

FORM PTO-1390 US DEPARTMENT OF COMMERCE REV. 5-93PATENT AND TRADEMARK OFFICE		ATTORNEYS DOCKET NUMBER <b>P01,0041</b>
<b>TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371</b>		U.S. APPLICATION NO. (if known, see 37 CFR 1.5) <b>09/786388</b>
INTERNATIONAL APPLICATION NO. <b>PCT/DE99/02753</b>	INTERNATIONAL FILING DATE <b>01 SEPTEMBER 1999</b>	PRIORITY DATE CLAIMED <b>02 SEPTEMBER 1998</b>
TITLE OF INVENTION <b>METHOD FOR DETERMINING A GRAPHIC STRUCTURE OF A TECHNICAL SYSTEM AND ARRANGEMENT AND SET OF ARRANGEMENTS FOR DETERMINING A GRAPHIC STRUCTURE</b>		
APPLICANT(S) FOR DO/EO/US <b>ERWIN THURNER</b>		
Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:		
<ol style="list-style-type: none"> <li>1. <input checked="" type="checkbox"/> This is a <b>FIRST</b> submission of items concerning a filing under 35 U.S.C. 371.</li> <li>2. <input type="checkbox"/> This is a <b>SECOND</b> or <b>SUBSEQUENT</b> submission of items concerning a filing under 35 U.S.C. 371.</li> <li>3. <input checked="" type="checkbox"/> This express request to begin national examination procedures (35 U.S.C. 371(f)) at any time rather than delay.</li> <li>4. <input checked="" type="checkbox"/> A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.</li> <li>5. <input checked="" type="checkbox"/> A copy of International Application as filed (35 U.S.C. 371(c)(2)) - drawings attached. <ol style="list-style-type: none"> <li>a. <input checked="" type="checkbox"/> is transmitted herewith (required only if not transmitted by the International Bureau).</li> <li>b. <input type="checkbox"/> has been transmitted by the International Bureau.</li> <li>c. <input type="checkbox"/> is not required, as the application was filed in the United States Receiving Office (RO/US)</li> </ol> </li> <li>6. <input checked="" type="checkbox"/> A translation of the International Application into English (35 U.S.C. 371(c)(2)) - drawings attached.</li> <li>7. <input checked="" type="checkbox"/> Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. §371(c)(3)) <ol style="list-style-type: none"> <li>a. <input type="checkbox"/> are transmitted herewith (required only if not transmitted by the International Bureau).</li> <li>b. <input type="checkbox"/> have been transmitted by the International Bureau.</li> <li>c. <input type="checkbox"/> have not been made; however, the time limit for making such amendments has NOT expired.</li> <li>d. <input checked="" type="checkbox"/> have not been made and will not be made.</li> </ol> </li> <li>8. <input type="checkbox"/> A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).</li> <li>9. <input checked="" type="checkbox"/> An oath or declaration of the inventor(s) (35 U.S.C. 371(c)(4)).</li> <li>10. <input checked="" type="checkbox"/> A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).</li> </ol> <p><b>Items 11. to 16. below concern other document(s) or information included:</b></p> <ol style="list-style-type: none"> <li>11. <input checked="" type="checkbox"/> An Information Disclosure Statement under 37 C.F.R. 1.97 and 1.98; (PTO 1449, Prior Art, Search Report, 04 References).</li> <li>12. <input type="checkbox"/> An assignment document for recording. A separate cover sheet in compliance with 37 C.F.R. 3.28 and 3.31 is included. (SEE ATTACHED ENVELOPE)</li> <li>13. <input checked="" type="checkbox"/> Amendment "A" Prior to Action and Appendix "A". <input type="checkbox"/> A SECOND or SUBSEQUENT preliminary amendment.</li> <li>14. <input checked="" type="checkbox"/> A substitute specification and substitute specification mark-up.</li> <li>15. <input type="checkbox"/> A change of address letter attached to the Declaration.</li> <li>16. <input checked="" type="checkbox"/> Other items or information: <ol style="list-style-type: none"> <li>a. <input checked="" type="checkbox"/> Submission of Drawings</li> <li>b. <input checked="" type="checkbox"/> EXPRESS MAIL #EL655301165US dated March 2, 2001</li> </ol> </li> </ol>		

U.S. APPLICATION NO. <b>09/786388</b> <small>(Unknown, see 37 C.F.R. 1.55)</small>		INTERNATIONAL APPLICATION NO. <b>PCT/DE99/02753</b>		ATTORNEY'S DOCKET NUMBER <b>P01,0041</b>	
---	--	--	--	---	--

17. <input checked="" type="checkbox"/> The following fees are submitted:  <b>BASIC NATIONAL FEE (37 C.F.R. 1.492(a)(1)(5):</b> Search Report has been prepared by the EPO or JPO \$860.00  International preliminary examination fee paid to USPTO (37 C.F.R. 1.482) \$690.00  No international preliminary examination fee paid to USPTO (37 C.F.R. 1.482) but international search fee paid to USPTO (37 C.F.R. 1.445(a)(2)) \$710.00  Neither international preliminary examination fee (37 C.F.R. 1.482) nor international search fee (37 C.F.R. 1.445(a)(2)) paid to USPTO \$1000.00  International preliminary examination fee paid to USPTO (37 C.F.R. 1.482) and all claims satisfied provisions of PCT Article 33(2)-(4) \$100.00  <div style="text-align: right;"><b>ENTER APPROPRIATE BASIC FEE AMOUNT =</b></div>				CALCULATIONS		PTO USE ONLY	
Surcharge of \$130.00 for furnishing the oath or declaration later than <input type="checkbox"/> 20 <input type="checkbox"/> 30 months from the earliest claimed priority date (37 C.F.R. 1.492(e)).				\$			
Claims	Number Filed	Number Extra	Rate				
Total Claims	14 - 20 =	0	X \$ 18.00	\$			
Independent Claims	02 - 3 =	0	X \$ 80.00	\$			
Multiple Dependent Claims			\$270.00 +	\$			
<b>TOTAL OF ABOVE CALCULATIONS =</b>				\$ 860.00			
Reduction by 1/2 for filing by small entity, if applicable. Verified Small Entity statement must also be filed. (Note 37 C.F.R. 1.9, 1.27, 1.28)				\$			
<b>SUBTOTAL =</b>				\$ 860.00			
Processing fee of \$130.00 for furnishing the English translation later than <input type="checkbox"/> 20 <input type="checkbox"/> 30 months from the earliest claimed priority date (37 CFR 1.492(f)). +				\$			
<b>TOTAL NATIONAL FEE =</b>				\$ 860.00			
Fee for recording the enclosed assignment (37 C.F.R. 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 C.F.R. 3.28, 3.31). \$40.00 per property +							
<b>TOTAL FEES ENCLOSED =</b>				\$ 860.00			
				Amount to be refunded	\$		
				charged	\$		

a. ☒ A check in the amount of \$860.00 to cover the above fees is enclosed.

b. ☐ Please charge my Deposit Account No. \_\_\_\_\_ in the amount of \$ \_\_\_\_\_ to cover the above fees. A duplicate copy of this sheet is enclosed.


c. ☒ The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 50-1519. **A duplicate copy of this sheet is enclosed.**

NOTE: Where an appropriate time limit under 37 C.F.R. 1.494 or 1.495 has not been met, a petition to revive (37 C.F.R. 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

**SEND ALL CORRESPONDENCE TO:**

**SCHIFF HARDIN & WAITE**  
**PATENT DEPARTMENT**  
**6600 Sears Tower**  
**233 South Wacker Drive**  
**Chicago, Illinois 60606-6473**

**CUSTOMER NUMBER 26574**

  
 SIGNATURE

Mark Bergner  
 NAME

45,877  
 Registration Number

BOX PCT  
IN THE UNITED STATES DESIGNATED/ELECTED OFFICE  
OF THE UNITED STATES PATENT AND TRADEMARK OFFICE  
UNDER THE PATENT COOPERATION TREATY--CHAPTER II

5

**PRELIMINARY AMENDMENT A**  
**PRIOR TO ACTION**

APPLICANT(S): ERWIN THURNER  
ATTORNEY DOCKET NO.: P01,0041  
INTERNATIONAL APPLICATION NO: PCT/DE99/02753  
INTERNATIONAL FILING DATE: 01 SEPTEMBER 1999  
INVENTION: METHOD FOR DETERMINING A GRAPHIC  
STRUCTURE OF A TECHNICAL SYSTEM AND  
ARRANGEMENT AND SET OF ARRANGEMENTS  
FOR DETERMINING A GRAPHIC STRUCTURE

10

Assistant Commissioner for Patents,  
Washington D.C. 20231

Sir:

15

Applicants herewith amend the above-referenced PCT application, and  
request entry of the Amendment prior to examination on the United States  
Examination Phase.

**IN THE SPECIFICATION**

20

Please cancel the code listings on pages 15-32—they are replicated in their  
entirety in the Appendix of the Substitute Specification.

**IN THE CLAIMS:**

**On substitute page 33:**

25

replace line 1 with --WHAT IS CLAIMED IS:--;

Please replace original claims 1-14 with the following rewritten claims 1-14,  
referring to the mark-ups in Appendix A.

1. (Amended) A method for determining a graphic structure of a technical  
system, comprising the steps of:

- 5
- a) selecting a graphic structure file from a set of a plurality of different graphic structure files, each graphic structure file containing indications of which elements can be selected to represent it in order to describe the structure of the technical system graphically;
  - b) selecting said elements in said graphic structure file in such a way that said technical system is described using said selected elements, and
  - c) representing said elements by an editor program into which said selected graph structure file has been integrated, which determines said graphic structure of said technical system.

10

2. (Amended) The method as claimed in claim 1, wherein said technical system is an electronic circuit.

15 3. (Amended) The method as claimed in claim 2, wherein said technical system is a piece of technical equipment.

4. (Amended) The method as claimed in claim 1, wherein said elements are graphic elements of a graphic which describe said technical system.

20 5. (Amended) The method as claimed in claim 1, further comprising the step of checking said graphic structure of said technical system for predefined structure rules.

25 6. (Amended) An arrangement for determining a graphic structure of a technical system, comprising:

- a) a memory in which a set of a plurality of different graphic structure files are stored, each said graphic structure file comprising indications of which elements can be selected to represent it in order to form a graphic;
  - b) a selector unit with which a graphic structure file can be selected from
- 30



said set of graph structure files;

- c) a processor configured to execute an editor program, said editor program being used to determine a graphic with elements of said selected graphic structure file via which a graphic structure is determined; and
- d) a representation component which is coupled to said editor program and with which a specific graph structure can be represented.

5  
10 7. (Amended) The arrangement as claimed in claim 6, wherein a structure of a technical system is described using the graph.

8. (Amended) The arrangement as claimed in claim 7, wherein said technical system is an electronic circuit.

15 9. (Amended) The arrangement as claimed in claim 7, wherein said technical system is a piece of technical equipment.

10. (Amended) The arrangement as claimed in claim 6, further comprising:

- a) a first subarrangement which comprises said memory; and
- b) a second subarrangement which is coupled to said first subarrangement and comprises:

said selector unit;  
said editor program; and  
said representation component.

25 11. (Amended) The arrangement as claimed in claim 10, wherein said first subarrangement and said second subarrangement are coupled to one another via a communications network.

30

12. (Amended) The arrangements as claimed in claim 10, wherein said structure of a technical system is described using a graphic.

13. (Amended) The arrangement as claimed in claim 12, wherein said  
5 technical system is an electronic circuit.

14. (Amended) The arrangement as claimed in claim 12, wherein said technical system is a piece of technical equipment.

### REMARKS


10

The present Amendment revises the specification and claims to conform to United States patent practice, before examination of the present PCT application in the United States National Examination Phase. Pursuant to 37 CFR 1.125 (b), applicants have concurrently submitted a substitute specification, excluding the  
15 claims, and provided a marked-up copy. All of the changes are editorial and applicant believes no new matter is added thereby. The amendment, addition, and/or cancellation of claims is not intended to be a surrender of any of the subject matter of those claims.

Early examination on the merits is respectfully requested.

20

Submitted by,

 (Reg. No. 45,877)  
Mark Bergner  
Schiff Hardin & Waite  
Patent Department  
25 6600 Sears Tower  
233 South Wacker Drive  
Chicago, Illinois 60606-6473  
(312) 258-5779  
30 Attorneys for Applicant  
**CUSTOMER NUMBER 26574**

Appendix A  
Mark Ups for Claim Amendments

This redlined draft, generated by CompareRite (TM) - The Instant Redliner, shows the differences between -  
original document : Q:\DOCUMENTS\YEAR 2001\P010041-THURNER-  
DETERMINING A GRAPHIC STRUCTURE\ORIGINAL CLAIMS.DOC  
and revised document: Q:\DOCUMENTS\YEAR 2001\P010041-THURNER-  
DETERMINING A GRAPHIC STRUCTURE\AMENDED CLAIMS.DOC

CompareRite found 87 change(s) in the text

Deletions appear as Overstrike text surrounded by []  
Additions appear as Bold-Underline text

1. **(Amended)** A method for determining a graphic structure of a technical system, **comprising the steps of:**

- a) **selecting a graphic** [a) in which a graph] structure file [is selected] from a set of a plurality of different [graph] **graphic** structure files, [a graph] **each graphic** structure file containing [in each case] indications of which elements can be selected to represent it in order to describe the structure of the technical system graphically[.];
- b) [in which] **selecting said** elements [are selected] **in said graphic structure file** in such a way that [the] **said** technical system is described using [the] **said** selected elements, and
- c) [in which the] **representing said** elements [are represented] by an editor program into which [the] **said** selected graph structure file has been integrated, [by] which [means the] **determines said** graphic structure of [the] **said** technical system [is determined].

2. **(Amended)** The method as claimed in claim 1, [in which the] **wherein said** technical system is an electronic circuit.

3. **(Amended)** The method as claimed in claim 2, [in which the] **wherein said** technical system is a piece of technical equipment.

4. (Amended) The method as claimed in ~~[one of claims 1 to 3, in which the]~~  
claim 1, wherein said elements are ~~[graph]~~ **graphic** elements of a ~~[graph]~~ **graphic**  
which describe ~~[the]~~ said technical system.

5. (Amended) The method as claimed in ~~[one of claims 1 to 4, in which the]~~  
claim 1, further comprising the step of checking said graphic structure of ~~[the]~~  
said technical system ~~[which is determined is checked]~~ for predefined structure  
rules.

6. (Amended) An arrangement for determining a ~~[graph]~~ **graphic** structure of  
a technical system, comprising:

- a) ~~[having]~~ a memory in which a set of a plurality of different ~~[graph]~~  
**graphic** structure files are stored, ~~[a graph]~~ each said graphic  
structure file ~~[containing in each case]~~ comprising indications of which  
elements can be selected to represent it in order to form a ~~[graph,]~~  
graphic;
- b) ~~[having]~~ a selector unit with which a ~~[graph]~~ **graphic** structure file can  
be selected from ~~[the]~~ said set of graph structure files~~[,]~~;
- c) ~~[having]~~ a processor ~~[which is]~~ configured ~~[in such a way that]~~ to  
execute an editor program ~~[can be executed, with which,]~~ said editor  
program ~~[a graph structure file selected from the set of graph structure~~  
~~files can be]~~ being used to determine a ~~[graph]~~ **graphic** with elements  
of ~~[the]~~ said selected ~~[graph]~~ **graphic** structure file~~[,] by]~~ via which  
~~[means the graph]~~ a graphic structure is determined; and[,]
- d) ~~[having]~~ a representation component which is coupled to ~~[the]~~ said  
editor program and with which ~~[the]~~ a specific graph structure can be  
represented.

7. (Amended) The arrangement as claimed in claim 6, ~~[in which]~~ wherein a

structure of a technical system is described using the graph.

8. **(Amended)** The arrangement as claimed in claim 7, ~~[in which the]~~  
wherein said technical system is an electronic circuit.

9. **(Amended)** The arrangement as claimed in claim 7, ~~[in which the]~~  
wherein said technical system is a piece of technical equipment.

10. **(Amended)** The arrangement as claimed in claim 6, further comprising:

- a) ~~[a) having]~~ a first subarrangement which ~~[has the]~~ comprises said  
memory; and;
- b) ~~[having]~~ a second subarrangement which is coupled to ~~[the]~~ said first  
subarrangement and ~~[has the following components:]~~ comprises:  
~~[the]~~ said selector unit;
- ~~[the]~~ said editor program; and;
- ~~[the]~~ said representation component.

11. **(Amended)** The arrangement as claimed in claim 10, ~~[in which the]~~  
wherein said first subarrangement and ~~[the]~~ said second subarrangement are  
coupled to one another ~~[by means of]~~ via a communications network.

12. **(Amended)** The ~~[set of]~~ arrangements as claimed in claim 10 ~~[or 11, in~~  
~~which a]~~, wherein said structure of a technical system is described using ~~[the~~  
~~graph.]~~ a graphic.

13. **(Amended)** The arrangement as claimed in claim 12, ~~[in which the]~~  
wherein said technical system is an electronic circuit.

14. **(Amended)** The arrangement as claimed in claim 12, ~~[in which the]~~  
wherein said technical system is a piece of technical equipment.

## SPECIFICATION

## TITLE

METHOD FOR DETERMINING A GRAPHIC STRUCTURE OF A TECHNICAL  
SYSTEM AND ARRANGEMENT AND SET OF ARRANGEMENTS FOR

5 DETERMINING A GRAPHIC STRUCTURE

## BACKGROUND OF THE INVENTION

## Field of the Invention

1 The invention relates to the selection of elements of a graph structure file in order to describe the structure of a technical system graphically.

10

## Description of the Related Art

2 It is known to describe different technical systems by means of a graphic structure. Such descriptions are known from, for example, product brochures for products provided by Zuken-Redac (e.g., Analysis Products, CAD Products, CAE  
15 Products, CAM Products, and Data Conversion Products—formerly available on September 22, 1998 at [http://www.redac.co.uk/prod\\_info/brochures/14a.html](http://www.redac.co.uk/prod_info/brochures/14a.html)) (the Zuken-Redac brochures), herein incorporated by reference, that disclose how, for a technical system such as an electronic circuit, the electrical circuit is determined in the form of a graphic structure with elements which describe an electronic circuit.

20 3 Elements of a graphic structure in the field of a circuit simulation are symbols which symbolize electronic components, for example, a resistor, a capacitor, an inductor, a transistor, an operational amplifier or other electronic components composed of these elements.

4 In the method and arrangement known from the Zuken-Redac brochures,  
25 elements for graphically describing an electronic circuit which are made available to a user by an editor program are selected in such a way that the "electronic circuit" constituting the technical system is described using the selected elements. The elements are represented by the editor program.

5 A graphic structure describes a graphic  $G (= V, E, \Psi)$  which has a finite, non-  
30 empty set  $V$  ( $v \in V$  designate nodes of the graphic  $G$ ), and a finite set  $E$  ( $e \in E$

designate edges of the graphic G). The nodes and edges of the graphic are logically combined by an incidence function  $\Psi$  which is formed according to the following rule:

$$\Psi: E \rightarrow \{\{i, j\} \mid i, j \in V\} \quad (1)$$

6 Each edge  $e$  of the set  $E$  of edges is assigned its two end places by the  
5 incidence function  $\Psi(e)$ .

7 Depending on the field of application, different types of nodes and edges may  
be provided in an editor program for describing a technical system. Nodes and  
edges to which an application-dependent semantic is assigned are generally  
designated as elements of the graphic in an editor program. A node element of a  
10 graphic is, for example in the editor program in the Zuken-Redac brochures, a  
symbol which symbolizes an electronic component of the electronic circuit. The  
edges can be used to describe weighted connections between the individual  
elements. Generally, the respective nodes and edges can be assigned a weight, a  
value or any desired text for information (textual information).

15 8 G. Chiola, G. Franceschinis, R. Gaeta and M. Ribaud, GreatSPN 1.7:  
Graphical Editor and Analyzer for Timed and Stochastic Petri Nets, Performance  
Evaluation, special issue on Performance Modeling Tools, 24 (1&2), pp. 47 - 68,  
November 1995 (Chiola), herein incorporated by reference, discloses an editor  
program for determining a Petri net. A Petri net is preferably used to analyze and  
20 design a closed-loop control system or an open-loop control system of a technical  
system, generally for describing system characteristics of a technical system. A  
graphic, which is illustrated in the form of a Petri net, has a place  $S$  or a transition  $T$   
as elements. A general overview of a Petri net and its basic elements can be found  
in G. Schmidt, Grundlagen der Regelungstechnik: Analyse und Entwurf linearer und  
25 einfacher nichtlinearer Regelungen sowie diskreter Steuerungen [Principles of  
control technology: analysis and design of linear and simple nonlinear closed-loop  
controls and discrete open-loop controls], second edition, Springer-Verlag  
[Publishing House], ISBN 3-540-17112-6, Berlin, pp. 320 - 328, 1991 (Schmidt),  
herein incorporated by reference.

30 9 A Petri net is generally a triplet

$$N := \langle S, T, F \rangle$$

where

(i)  $S = \{ s_1, s_2, \dots, s_n \}$

Set of places

(ii)  $T = \{ t_1, t_2, \dots, t_m \}$

Set of transitions

(iii)  $S \cap T = \emptyset$

S and T disjunctive

5

(the node set is

composed of S and T)

(iv)  $F \subseteq (S \times T) \cup (T \times S)$

Flow relation

10 A particular disadvantage with the known methods and arrangements is the  
10 fact that in each case elements of a graphic which are provided only for a specific  
application are made available as a function of the application in order to determine  
the graphic structure of a technical system. Thus, with the editor program from the  
Zuken-Redac brochures, only a selection of the elements can be made to describe  
an electronic circuit, and in the case of the editor program from Chiola, only a  
15 selection of elements can be made to describe a Petri net.

11 Such a known editor program is thus extremely inflexible in a situation in  
which a user wishes to use different types of a graphic structure to describe a  
technical system. In this type of program, it is necessary to develop for each specific  
application a separate editor program which is adapted to the application, something  
20 which entails considerable development costs.

### SUMMARY OF THE INVENTION

12 The invention is therefore based on providing a method for determining a  
graphic structure of a technical system, and an arrangement and a set of a plurality  
25 of arrangements for determining a graphic structure which has improved flexibility in  
comparison with the known methods and arrangements.

13 The problem is solved by a method for determining a graphic structure of a  
technical system (which may be an electronic circuit or a piece of technical  
equipment) has the following steps:

30 14 a) a graphic structure file is selected from a set of a plurality of different  
graphic structure files, a graphic structure file containing, in each case, indications of



which elements can be selected to represent the graphic structure file in order to describe the structure of the technical system graphically,

15     b) elements are selected in such a way that a technical system is described using the selected elements, and

5     16     c) the elements are represented by an editor program into which the selected graphic structure file has been integrated, via which the graphic structure of the technical system is determined.

17     The problem is also solved by an arrangement for determining a graphic structure has the following features:

10    18     a) a memory in which a set of a plurality of different graphic structure files are stored, a graphic structure file containing, in each case, indications of which elements can be selected to represent it in order to form a graphic,

19     b) a selector unit with which a graphic structure file can be selected from the set of graphic structure files,

15    20     c) a processor configured to execute an editor program, with which editor program a graphic structure file selected from the set of graphic structure files can be used to determine a graphic with elements of the selected graphic structure file, by which means the graphic structure is determined, and

21     d) a representation component which is coupled to the editor program and  
20     with which the specific graphic structure can be represented.

22     In the inventive method, the elements may be graphic elements of a graphic which describes the technical system. Also, a further step of checking the graphic structure of the technical system for predefined structure rules may be provided as well.

25    23     A set of a plurality of arrangements for determining a graphic structure has:

24     a) a first arrangement which has a memory in which a set of a plurality of different graphic structure files are stored, a graphic structure file containing in each case indications of which elements can be selected to represent it in order to form a graphic, and

30    25     b) a second arrangement which is coupled to the first arrangement and has the following components:

26           - a selector unit with which a graphic structure file can be selected from  
the set of graphic structure files,

27           - an editor program with which a graphic structure file selected from the  
set of graphic structure files can be used to determine a graphic with elements, of  
5 the selected graphic structure file, via which the graphic structure is determined, and

28           - a representation component which is coupled to the editor program  
and with which the specific graphic structure can be represented.

29       The invention discloses a method which is very flexible in comparison with the  
known methods and arrangements, and a very flexible arrangement for determining  
10 a graphic structure which can be adapted to new application scenarios in a quick  
and easy way and can be adapted more satisfactorily to existing application  
scenarios.

30       In this way, different types of structures which can be represented as a  
graphic can be processed flexibly, cost-effectively, and easily with the inventive  
15 method or arrangement.

31       These inventive aspects are described in more detail below.

32       The technical system is preferably an electronic circuit or a piece of technical  
equipment. The elements are preferably graphic elements of a graphic which  
describe the technical system.

20 33       In a further refinement there is provision for the graphic structure of the  
technical system which is determined to be checked for predefined structure rules. In  
this way, it is possible to check a structure of the technical system determined by a  
user for predefined structure rules, which ensures that the structure rules for the  
respective technical system are complied within terms of its graphic structure.

25 34       An exemplary structure rule could be, for example, in a Petri net, the fact that  
a place must always follow a transition, and vice versa. If this is not the case, within  
the scope of this development, the disclosure is made during checking of the graphic  
structure of a Petri net that the corresponding structure rule is infringed.

## BRIEF DESCRIPTION OF THE DRAWINGS

35 An exemplary embodiment of the invention is illustrated in the figures and explained in more detail below.

36 Figure 1 is a schematic diagram showing an arrangement according to a first  
5 exemplary embodiment;

37 Figure 2 is a pictorial diagram of a representation component with a graphic structure with elements of a Petri net;

38 Figure 3 is a pictorial diagram of a representation component with a graphic structure with elements which describe an electronic circuit;

10 39 Figure 4 is a flowchart in which the method steps of the method according to an exemplary embodiment are represented; and

40 Figure 5 is a block diagram of a set of a plurality of arrangements which, according to a second exemplary embodiment, are coupled to one another via a communications network.

15

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

41 Fig. 1 shows an arrangement 100 with a set 101 of a plurality of different graphic structure files 102, 103, 104, 105. Each graphic structure file 102, 103, 104, 105 is embodied as a file which can be linked dynamically (dynamic link library).

20 42 A user 106 selects a graphic structure file 102, 103, 104, 105 using a selection component 108 (keyboard and/or computer mouse) which is connected to an editor program 107.

43 The selected graphic structure file, in this exemplary embodiment a first graphic structure file 103, is dynamically integrated into the editor program 107.

25 44 After integration into the editor program 107, a set 111 of selectable elements 112, 113, 114, which are defined in the first graphic structure file 103 as selectable elements for determining a further described graphic is displayed to the user 106 on a screen 110 via a representation component 109 which is connected to the editor program 107. In addition, in this exemplary embodiment, according to the first  
30 graphic structure file 103, a first check program 115 and a second check program

116 are integrated into the editor program 107 and made available to the user 106 for selection.

45 Each graphic structure file 102, 103, 104, 105 has, in each case, a set of selectable elements for the respective type of graphics, in each case a graphic  
5 structure file being provided for one type of graphic. In addition, each graphic structure file 102, 103, 104, 105 may respectively contain a specific check program which is integrated into the respective graphic structure file 102, 103, 104, 105.

46 After the first graphic structure file 103 has been integrated, elements of the graphic are selected by the user 106 and connected to one another so that a graphic  
10 is determined which is stored in the form of a predefined intermediate language 117 in a memory 118.

47 In addition, Fig. 1 symbolically represents that the user 106 stores a plurality of structures 119, 120, 121, 122, 123 for describing different graphics, these structures relating to the type of graphic predefined by the first graphic file 103. The  
15 first graphic structure file 103 makes available elements which make possible a graphic in the form of a Petri net 201 (see Fig. 2).

48 Fig. 2 shows the representation component 200 which is presented to the user 106 in the form of a screen surface. The screen surface 200 has a menu list  
20 202 with different selectable options ("File", "Edit", "Object", "View", "Tools", "Settings", "Help"). Menu items are made available to the user by means of individual selectable elements using an immediate access bar 203 by making a single, direct selection of an element.

49 In addition, a processing bar 204 is represented with selectable options for determining the graphic. Thus, in the first graphic structure file 103, a first selection  
25 element 205 is provided with which it is possible to select and process objects represented on the screen. The selection and processing of specific elements for a Petri net 201 is made available to the user 106 via a set 206 of further selector elements which will be explained in more detail below.

50 A second selector element 207 is described by an empty rectangle and  
30 symbolizes a time-specific transition. A third selector element 208 symbolizes a timeless transition, which is represented as a selected transition element 220, 221

and 222 in the Petri net 201. A fourth selector element 209 symbolizes an edge which is a directed edge in this exemplary embodiment. A fifth selector element 210 symbolizes a forbidden edge which is designated in accordance with the structure rules of a Petri net 201. A sixth selector element 211 symbolizes a place where, in  
5 each case, a place element 223, 224, 225, 226 is represented in the Petri net 201. The place elements 223, 224, 225 and 226 are connected to the transition elements 220, 221, 222 via edges 227, 228, 229, 230, 231 and 232. A seventh selector element 212 symbolizes the possibility of combining a plurality of elements of the Petri net to form a composite element. An eighth selector element 213 symbolizes  
10 an input of the Petri net 201 and a ninth selector element 214 symbolizes an output of a Petri net 201.

51 The edges and the individual nodes, i.e., the elements of the Petri net 201, are assigned textual information 251, 252, 253, 254, 255, 256, 257, 258, 259, 260 and 261. In this way it is possible to assign an additional textual description to the  
15 individual elements.

52 If a second graphic structure file 104 is integrated into the editor program 107, the second graphic structure file 104 making available elements of an electronic circuit, and thus a graphic of an electronic circuit, a screen mask represented in Fig. 3 with a set of selector elements set up for the circuit simulation is produced.

20 53 The same designations are used in Fig. 3 for the same elements displayed on the screen as represented in Fig. 2.

54 A set 301 of selector elements which are specifically for describing a graphic of an electronic circuit contain :

- a tenth selector element 302 which symbolizes an electronic resistor,
- 25 • an eleventh selector element 303 which symbolizes an electronic capacitor,
- a twelfth selector element 304 which symbolizes an inductor,
- a thirteenth selector element 305 symbolizing a transistor,
- a fourteenth selector element 306 symbolizing an operational  
30 amplifier,

• a fifteenth selector element 307 symbolizing a non-directed edge,  
and

• a sixteenth selector element 308 symbolizing a power source.

55 An electronic circuit 110 is determined by the user 106 and has, in this  
5 exemplary embodiment, a power source 311, electronic resistors 312, 313,  
electronic capacitors 314 and 315 and an operational amplifier 316 which are each  
connected to one another via edges 317. In addition, a ground terminal 318 is  
illustrated in Fig. 3. The individual circuit elements are assigned textual information  
319, 320, 321, 322, 323, 324, 325, 326 for further explaining the electronic circuit  
10 310.

56 Fig. 4 shows the inventive method steps. In a first step (step 401) a graphic  
structure file 102, 103, 104, 105 is selected from a set 101 of graphic structure files  
102, 103, 104, 105. In a second step (step 402), a selection is made of elements  
which are available in accordance with the graphic structure file 102, 103, 104, 105  
15 which was selected in the previous step (step 401). The selected elements are  
illustrated by the editor program 107 in a further step (step 403).

57 Fig. 5 shows a first computer 500 with a memory 502 and a processor 503  
which are each connected to one another via a bus 504 and to an input/output  
interface 501. The first computer 500 is connected to a screen 505, a keyboard 506,  
20 and a computer mouse 507 via the input/output interface 501.

58 In addition, the first computer 500 is connected to further computers 510, 520,  
530, 540 and 550 via a communications network 560, in the exemplary embodiment,  
an ISDN network (Integrated Services Digital Network).

59 The set 101 of graphic structure files 102, 103, 104, 105 is stored in the first  
25 computer 500. The further computers 510, 520, 530, 540 and 550 each also have a  
processor 513, 523, 533, 543 and 553 and each have a memory 512, 522, 532, 542  
and 552. In each case the processor 513, 523, 533, 543 and 553 and the memory  
512, 522, 532, 542 and 552 are connected to the communications network via, in  
each case, a bus 514, 524, 534, 544 and 554 via an input/output interface 511, 521,  
30 531, 541 and 551. In addition, the further computers 510, 520, 530, 540 and 550 are

each connected to a screen 515, 525, 535, 545 and 555, to a keyboard 516, 526, 536, 546 and 556 and to a computer mouse 517, 527, 537, 547 and 557.

60 An editor program 508, 518, 528, 538, 548, 558 is stored in each computer 500, 510, 520, 530, 540 and 550. A graphic structure file 102, 103, 104, 105 is  
5 selected by a user of a further computer 510, 520, 530, 540 and 550, and requested from the first computer 500 with a request message 570. The first computer 500 transmits the selected graphic structure file 102, 103, 104, 105 in a reply message 580 to the further computer 510, 520, 530, 540 and 550 requesting the graphic structure file 102, 103, 104, 105.

10 61 The requesting further computer 510, 520, 530, 540 and 550 has thus received the requested graphic structure file 102, 103, 104, 105, and it integrates it into its editor program 518, 528, 538, 548, 558, as described in the first exemplary embodiment.

62 A number of alternatives to the exemplary embodiments described above are  
15 illustrated as follows: The type of elements which are made available by a graphic structure file is generally freely selectable and depends only on the respective type of graphic to be determined. The technical system can, for example, also be a piece of technical equipment whose characteristics or structure can be described by the graphic. The editor program and the graphic illustrated with the editor program can  
20 be used as part of a simulation of the technical system.

63 Three files are provided in the Appendix which implement the exemplary embodiments written in the C/Java programming language. These files are: 1) an initialization file, 2) a load file, and 3) a toolbar file.

64 The above-described method and arrangement are illustrative of the  
25 principles of the present invention. Numerous modifications and adaptations will be readily apparent to those skilled in this art without departing from the spirit and scope of the present invention.

## ABSTRACT

65 A graphic structure file is selected from a set of a plurality of different graphic  
structure files. A graphic structure file contains in each case indications of which  
5 elements can be selected to represent it in order to describe the structure of the  
technical system graphically. Elements are selected in such a way that the selected  
elements describe the technical system, and the elements are represented by an  
editor program into which the selected graphic structure file has been integrated.



## APPENDIX CODE LISTINGS

### 1. Initialization file:

5

```

package interfaces;

import java.io.*;
import java.util.*;
import java.awt.*;

import etc.*;
import elements.*;
import mmi.*;
import tools.*;

public class Initialisierung {
    GraphEditor editor;
    // Der hat die Tokens aus der
    // Datei
    StreamTokenizer token;
    // Hier kommen alle erlaubten
    // Knoten und Kanten aus der
    // .lgc Datei rein.
    // Die Einträge werden mit den
    // Namen der Objekte referenziert
    Hashtable gobjekte;
    // Die aktuelle .lgc Datei
    //String configFile;
    // steht jetzt bei den Einstel-
    // lungen
    /**
     * Hier stehen alle Attribute
     * drin.
     */
    Hashtable attributNamen;
    /**
     * hier kommen die Einträge für
     * das Menü Tools
     * hinein.
     */
    Hashtable tools;

    public Initialisie-
    rung(GraphEditor editor) {
        this.editor = editor;
        gobjekte = new Hashtable();
        attributNamen = new Has-
        htable();
        tools = new Hashtable();
    }

    /**
     * Diese Methode würde die er-
     * ste Initialisierungsdatei
     *
     * einlesen für die Einstellu-
     * gen der Farben, Schriften...
     * Aber ich darf leider nicht.
     */
    /**
     public void readFirst(String
     name) {
        String configFile = new
        String(name);
        int c;
        //Properties properties = new
        Properties();
        //properties = Sy-
        stem.getProperties();
        //filename = new String("..")
        + proper-
        ties.getProperty("file.separator"
        ) + configFile);
        try {
            File file = new
            File(configFile);
            //FileInputStream in = new
            FileInputStream(file);
            FileReader in = new File-
            Reader(file);
            token = new StreamToken-
            izer(in);

            //Einstellen der Optionen
            für token
            to-
            ken.eolIsSignificant(true);
            token.quoteChar('"');
            //token.quoteChar('\\');
            //token.quoteChar('{');
            token.quoteChar('}');

            //Überlese { und , und ;
            to-
            ken.whitespaceChars('(','(');
            to-
            ken.whitespaceChars(',',',');
            to-
            ken.whitespaceChars(';',';');

            boolean fertig = false;
            while (!fertig) {
                switch
                (c=token.nextToken()){

```

```

        case StreamTokeniz-
zer.TT_EOF:
            fertig= true;
            break;
        case StreamTokeniz-
zer.TT_WORD:
            if
(token.sval.equals("DATAPATH")) {
c=token.nextToken();
            if (c == '"') {
                Sy-
stem.out.println("DATAPATH " +
token.sval);
            }
            break;
        }
        if
(token.sval.equals("DATAFILTER"))
{
c=token.nextToken();
            if (c == '"') {
                Sy-
stem.out.println("DATAFILTER " +
token.sval);
            }
            break;
        }
        if
(token.sval.equals("FILELIST")) {
            while (c != '"')
{
c=token.nextToken();
            if (c ==
'"') {
                edi-
tor.getMenueleiste().addFileToMen-
u(token.sval);
            }
            break;
        }
        if
(token.sval.equals("COLORS")) {
            while (c != '"')
{
c=token.nextToken();
            if (c == Stre-
amTokenizer.TT_WORD) {
                String aus-
wahl = token.sval;
c=token.nextToken();
//System.out.print("Wert1 " + to-
ken.nval);
                int r =
(int)token.nval;

```

```

c=token.nextToken();
//System.out.print("Wert2 " + to-
ken.nval);
                int g =
(int)token.nval;
c=token.nextToken();
//System.out.println("Wert3 " +
token.nval);
                int b =
(int)token.nval;
//System.out.flush();
                ueberge-
be(auswahl,r,g,b);
            }
            break;
        }
        if
(token.sval.equals("FONTS")) {
            while (c != '"')
{
c=token.nextToken();
            if (c == Stre-
amTokenizer.TT_WORD) {
                String aus-
wahl = token.sval;
//System.out.print("FONT " + to-
ken.sval);
c=token.nextToken();
                String font-
name = token.sval;
//System.out.print(" NAME " + to-
ken.sval);
c=token.nextToken();
                String style
= token.sval;
//System.out.print(" STYLE " +
token.sval);
c=token.nextToken();
                int size
=(int) token.nval;
                ueberge-
be(auswahl,fontname,style,size);
//System.out.println(" SIZE " +
token.nval);
            }
            break;
        }
}

```

```

        if
(token.sval.equals("SHORTCUTS"))
{
    while (c != ' ')
    {
        c=token.nextToken();
        if (c == '"')
        {
            String
mpunkt = token.sval;

//System.out.print("MENUPUNKT " +
token.sval);

c=token.nextToken();
String icon1
= token.sval;

//System.out.print("ICON1 " + to-
ken.sval);

c=token.nextToken();
String icon2
= token.sval;

//System.out.println("ICON2 " +
token.sval);

        edi-
tor.getShortcutableiste().addShortB
utton();
    }
    break;
}
if
(token.sval.equals("ACCELERATOR"))
{
    while (c != ' ')
    {
        c=token.nextToken();
        if (c == '"') {
            String la-
bel = token.sval;

//System.out.print("MENUPUNKT " +
token.sval);

c=token.nextToken();
        if (c ==
StreamTokenizer.TT_WORD) {
            char cut =
token.sval.charAt(0);

//System.out.println(" TASTEN " +
cut);

        edi-
tor.getMenueiste().addShortcutT
oVector(label, cut);
    }
}

```

```

    }
    break;
}
if
(token.sval.equals("WINDOWSIZE"))
{
    c=token.nextToken();
    int x
=(int)token.nval;

c=token.nextToken();

c=token.nextToken();
    int y
=(int)token.nval;

//size.setSize(x,y);
    break;
}
if
(token.sval.equals("WINDOWPOSITIO
N")) {
    c=token.nextToken();
    int x
=(int)token.nval;

c=token.nextToken();

c=token.nextToken();
    int y
=(int)token.nval;

//location.setSize(x,y);
    break;
}
if
(token.sval.equals("AUTHOR")) {
    c=token.nextToken();
    if (c == '"') {
        Sy-
stem.out.println("AUTHOR " + to-
ken.sval);
    }
    break;
}
if
(token.sval.equals("TOOLS")) {
    while (c != ' ')
    {
        c=token.nextToken();
        if (c == '"')
        {
            String pfad
=new String(token.sval);

//System.out.println("TOOL " +
token.sval);

```

```

c=token.nextToken();
        String fileName = new String(token.sval);

//System.out.println("TOOL " +
token.sval);

c=token.nextToken();
        String text
=new String(token.sval);

//System.out.println("TOOL " +
token.sval);

        editor.getMenueLeiste().addToolToVec
tor(pfad, fileName, text);
    }
    }
    break;
} else
    break;

default:
}
}

in.close();
System.out.flush();
System.out.println("EINLESEN
DER DATEI " + configFile + "
FERTIG");

} catch
(FileNotFoundException e) {
    System.err.println("con-
figFile + " is not found");
} catch (IOException e) {
    e.printStackTrace();
}
} //read first
*/

/**
 * Diese Methode liest eine
Toolbar ein.
 * Sie benötigt den Pfad zur
Datei und den Dateinamen.
 */
public void readSecond(String
lgcPath, String datei) {
    String configFile = new
String(lgcPath + datei);
    int c;
    try {
        File file = new
File(configFile);
        FileReader in = new File-
Reader(file);
        token = new StreamToken-
izer(in);

```

```

//Einstellen der Optionen
für token
to-
ken.eolIsSignificant(false);
token.quoteChar('');
//token.quoteChar('\\');
//token.quoteChar('{');
token.quoteChar('');

//Überlese ( und , und ;
to-
ken.whitespaceChars('(','(');
to-
ken.whitespaceChars(','','');
to-
ken.whitespaceChars(';','');

boolean fertig = false;
while (!fertig) {
    switch
(c=token.nextToken()){
        case StreamToken-
izer.TT_EOF:
            fertig= true;
            break;
        case StreamToken-
izer.TT_WORD:
            if
(token.sval.equals("TOOLBAR")) {
                Sy-
stem.out.println("Lese Toolbar");
                readTool-
bar(lgcPath);
                break;
            }
            if
(token.sval.equals("MENU")) {
                Sy-
stem.out.println("Lese Menue");
                readMenu();
                break;
            }
            if
(token.sval.equals("ANALYSISBAR"))
            {
                Sy-
stem.out.println("Lese Analyse-
Bar");
                readAnalyse();
                break;
            }
            if
(token.sval.equals("SHORTCUTS"))
            {
                Sy-
stem.out.println("Lese Short-
cuts");
                readShorts();
                break;
            }

```

```

        }
        if
(token.sval.equals("ACCELERATOR")
) {
            Sy-
stem.out.println("Lese Accelerator");
            readAccel();
            break;
        }
        default:
    }

    in.close();
    System.out.flush();
    System.out.println("EINLESEN
DER DATEI " + configFile + "
FERTIG!");
    //und wichtig für die Anzei-
ge:
    setLayer();
    setAttributNames();
    } catch
(FileNotFoundException e) {
        System.err.println( con-
figFile + " is not found");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void readToolbar(String
lgcPath) {
    int c = '{';
    gobjekte.clear();
    //System.out.println("Jetzt
kommt die Toolbar");
    try {
        while (c != '}') {
            switch
(c=token.nextToken()){
                case StreamToken-
zer.TT_WORD:
                    if
(token.sval.equals("NODE")) {
//System.out.println("Lese Kno-
ten");
                        readNode(lgcPath);
                        break;
                    }
                    if
(token.sval.equals("EDGE")) {
//System.out.println("Lese Kan-
te");
                        readEdge(lgcPath);
                        break;
                    }
                    default:
                }
            }
}

```

```

        //c=token.nextToken();
        //System.out.println("IN
der TOOLBAR " + c );
    } catch (IOException e) {
        e.printStackTrace();
    }
    //System.out.println("Fertig
Toolbar");
}

private void readNode(String
lgcPath) {
    int c = '{';
    String typname = new
String();
    String image = new String();
    Vector ecken = new Vector();
    Vector konnektoren = new Vec-
tor();
    Vector konnektorNamen = new
Vector();
    Attribute attribute = new
StandardAttribute();
    Color color = new Co-
lor(255,255,255);
    //System.out.println("Ein
Knoten");
    try {
        while (c != '}') {
            switch (c) {
                case StreamToken-
zer.TT_WORD:
                    // Wird nicht mehr be-
notigt
                    // if
(token.sval.equals("TYPE")) {
                        //
c=token.nextToken();
                        // Sy-
stem.out.println("Lese TYPE" +
token.sval);
                        // break;
                        // }
                        if
(token.sval.equals("NAME")) {
c=token.nextToken();
                            typname = new
String(token.sval);
                            // Sy-
stem.out.println("Lese NAME" +
typname);
                            break;
                        }
                        if
(token.sval.equals("ATTRIBUTES"))
{
                            attribute = new
StandardAttribute();

```

```

        while
        ((c=token.nextToken()) != ' ') {
            String aname =
            new String(token.sval);
            c = to-
            ken.nextToken();
            String wert = new
            String(token.sval);
            attribu-
            te.addAttribute(aname,wert,true);
            attribuNa-
            men.put(aname,aname);
            // Sy-
            stem.out.println("Lese Attribut-
            te" + attribute);
        }

        break;
    }
    if
    (token.sval.equals("IMAGE")) {

        c=token.nextToken();
        image = new
        String(token.sval);
        // Sy-
        stem.out.println("Lese IMAGE" +
        image);

        break;
    }
    if
    (token.sval.equals("FILLEDPOLYGON
    ")); {
        ek-
        ken.removeAllElements();
        int x,y;
        while
        ((c=token.nextToken()) != ' ') {
            x =
            (int)token.nval;

            c=token.nextToken();
            y =
            (int)token.nval;
            ek-
            ken.addElement(new Point(x,y));
            // Sy-
            stem.out.println("Lese POLYGON" +
            ecken);
        }
        // jetzt sollten
        alle Daten da sein, und es
        // kann ein Knoten-
        prototyp erzeugt werden.
        GraphObjekt knoten =
        new FilledPolygonKnoten(typname,
        ecken,
        konnektoren,
        konnektorNamen,

```

```

        attribute);
        kno-
        ten.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
        color);
        // Erzeuge Button
        mit Werkzeug für Werkzeugleiste
        ToolButton b = new
        ToolButton(lgcPath + "images/" +
        image,
        typename,
        new KnotenTool(editor,typename),
        editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        // Eintrag in die
        Hashtabelle
        gobjek-
        te.put(typname,knoten);
        // Sy-
        stem.out.println("In Hashtabelle:
        " + gobjekte);

        break;
    }
    if
    (token.sval.equals("POLYGON")) {
        ek-
        ken.removeAllElements();
        int x,y;
        while
        ((c=token.nextToken()) != ' ') {
            x =
            (int)token.nval;

            c=token.nextToken();
            y =
            (int)token.nval;
            ek-
            ken.addElement(new Point(x,y));
            // Sy-
            stem.out.println("Lese POLYGON" +
            ecken);
        }
        // jetzt sollten
        alle Daten da sein, und es
        // kann ein Knoten-
        prototyp erzeugt werden.
        GraphObjekt knoten =
        new PolygonKnoten(typname,
        ecken,
        konnektoren,
        konnektorNamen,

```

```

attribute);
        kno-
ten.setColor(color);
        // Sy-
stem.out.println("Setze Farbe " +
color);
        // Erzeuge Button
mit Werkzeug für Werkzeugleiste
        // Der Button greift
über den typnamen auf den richti-
gen
        // Knoten zu.
        ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typename,

new KnotenTool(editor,typename),

editor.getToolBar());
        edi-
tor.getToolBar().addToolButton(b)
;
        // Eintrag in die
Hashtabelle
        gobjek-
te.put(typname,knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

        break;
    }
    if
(token.sval.equals("FILLEDOVAL"))
{
        int breite=10;
        int hoehe=10;
        while
((c=token.nextToken()) != ' '){
            breite =
(int)token.nval;
c=token.nextToken();
            hoehe =
(int)token.nval;
            // Sy-
stem.out.println("Lese OVAL_FILL"
+ token.nval);
        }
        // jetzt sollten
alle Daten da sein, und es
        // kann ein Knoten-
prototyp erzeugt werden.
        GraphObjekt knoten
= new FilledOvalKnoten(typname,

hoehe,

breite,

```

```

konnektoren,

konnektorNamen,

attribute);
        kno-
ten.setColor(color);
        // Sy-
stem.out.println("Setze Farbe " +
color);
        // Erzeuge Button
mit Werkzeug für Werkzeugleiste
        ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typename,

new KnotenTool(editor,typename),

editor.getToolBar());
        edi-
tor.getToolBar().addToolButton(b)
;
        // Eintrag in die
Hashtabelle
        gobjek-
te.put(typname,knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

        break;
    }
    if
(token.sval.equals("OVAL")) {
        int breite=10;
        int hoehe=10;
        while
((c=token.nextToken()) != ' '){
            breite =
(int)token.nval;
c=token.nextToken();
            hoehe =
(int)token.nval;
            // Sy-
stem.out.println("Lese OVAL" +
token.nval);
        }
        // jetzt sollten
alle Daten da sein, und es
        // kann ein Knoten-
prototyp erzeugt werden.
        GraphObjekt knoten
= new OvalKnoten( typename,

hoehe,

breite,

```

```

konnektoren,
konnektorNamen,
attribute);
        kno-
ten.setColor(color);
        // Sy-
stem.out.println("Setze Farbe " +
color);
        // Erzeuge Button
mit Werkzeug für Werkzeugleiste
        ToolButton b = new
ToolButton(lgcPath + "images/" +
image,
typname,
new KnotenTool(editor, typname),
editor.getToolBar());
        edi-
tor.getToolBar().addToolButton(b);
        // Eintrag in die
Hashtabelle
        gobjek-
te.put(typname, knoten);
//System.out.println("In Hashta-
belle: " + gobjekte);
        break;
    }
    if
(token.sval.equals("CONNECTORS"))
    {
        konnekto-
ren.removeAllElements();
        int x,y;
        String name;
        while
        ((c=token.nextToken()) != '|') {
            x =
(int)token.nval;
c=token.nextToken();
            y =
(int)token.nval;
c=token.nextToken();
            name = to-
ken.sval;
            konnekto-
ren.addElement(new Point(x,y));
            konnektorNa-
men.addElement(name);
            // Sy-
stem.out.println("Lese Konnekto-
ren" + konnektoren);

```

```

        // Sy-
stem.out.println("Die Namen: " +
konnektorNamen);
        }
        break;
    }
    if
(token.sval.equals("COLOR")) {
c=token.nextToken();
//System.out.println("Lese COLOR"
+ token.nval);
        int r =
(int)token.nval;
c=token.nextToken();
//System.out.println("Lese COLOR"
+ token.nval);
        int g =
(int)token.nval;
c=token.nextToken();
//System.out.println("Lese COLOR"
+ token.nval);
        int b =
(int)token.nval;
        color = new Co-
lor(r,g,b);
        break;
    }
    default:
    } //switch
c=token.nextToken();
    // Sy-
stem.out.println("NAECHSTES
TOKEN" + token.sval);
    } //while
    //c=token.nextToken();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // System.out.println("Bende
readNode");
    } //readNode

    private void readEdge(String
lgcPath) {
        // System.out.println("Eine
Kante");
        int c = '|';
        String typname = new
String();
        String image = new String();
        Attribute attribute = new
StandardAttribute();

```



```

        Color color = new Color(255,255,255);
        try {
            while (c != '}') {
                switch (c){
                    case StreamTokenizer.TT_WORD:

                        if
                        (token.sval.equals("NAME")) {
                            c=token.nextToken();
                            typename = new
                                String(token.sval);
                            // Sy-
                            stem.out.println("Lese NAME" +
                                typename);
                            break;
                        }
                        if
                        (token.sval.equals("ATTRIBUTES"))
                        {
                            attribute = new
                                StandardAttribute();
                            while
                                ((c=token.nextToken()) != '}') {
                                    String aname =
                                        new String(token.sval);
                                    c = token.
                                        nextToken();
                                    String wert = new
                                        String(token.sval);
                                    attribute.
                                        addAttribut(aname,wert,true);
                                    attributNa-
                                        men.put(aname,aname);
                                    // Sy-
                                    stem.out.println("Lese Attribut-
                                        te" + attribute);
                                }
                            break;
                        }
                        if
                        (token.sval.equals("IMAGE")) {
                            c=token.nextToken();
                            image = new
                                String(token.sval);
                            // Sy-
                            stem.out.println("Lese IMAGE" +
                                image);
                            break;
                        }
                        if
                        (token.sval.equals("ARROW")) {
                            int radius = 10;
                            int winkel = 10;
                            while
                                ((c=token.nextToken()) != '}') {
                                    radius =
                                        (int)token.nval;

```

```

--
        c=token.nextToken();
        winkel =
            (int)token.nval;
        // Sy-
        stem.out.println("Lese Arrow" +
            radius+ winkel);
        }
        // jetzt sollten
        alle Daten da sein, und es
        // kann ein Kanten-
        prototyp erzeugt werden.
        GraphObjekt kante =
            new PfeilKante(typname,
                radius,
                winkel,
                attribute);
        kan-
        te.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
            color);
        // Erzeuge Button
        mit Werkzeug für Werkzeugleiste
        ToolButton b = new
            ToolButton(lgcPath + "images/" +
                image,
                typname,
                new KantenTool(editor, typname),
                editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        ;
        // Eintrag in die
        Hashtabelle
        gobjek-
        te.put(typname,kante);
        //System.out.println("In Hashta-
        belle: " + gobjekte);
        break;
    }
    if
    (token.sval.equals("POINT")) {
        int durch = 10;
        while
            ((c=token.nextToken()) != '}') {
                durch =
                    (int)token.nval;
                // Sy-
                stem.out.println("Lese Point" +
                    durch);
            }
        // jetzt sollten
        alle Daten da sein, und es

```

```

        // kann ein Kanten-
        // prototyp erzeugt werden.
        GraphObjekt kante =
        new KreisKante(typname,

        durch,

        attribute);
        kan-
        te.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
        color);
        // Erzeuge Button
        // mit Werkzeug für Werkzeugleiste
        ToolButton b = new
        ToolButton(lgcPath + "images/" +
        image,

        typname,

        new KantenTool(editor, typname),

        editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        // Eintrag in die
        // Hashtabelle
        gobjek-
        te.put(typname, kante);
        //System.out.println("In Hashta-
        //belle: " + gobjekte);

        break;
    }
    if
    (token.sval.equals("NOEND")) {
        while
        ((c=token.nextToken()) != ' '){
            // durch =
            (int)token.nval;
            // Sy-
            stem.out.println("Lese Point" +
            durch);
        }
        // jetzt sollten
        // alle Daten da sein, und es
        // kann ein Kanten-
        // prototyp erzeugt werden.
        GraphObjekt kante =
        new StandardKante(typname,

        attribute);
        kan-
        te.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
        color);

```

```

        // Erzeuge Button
        // mit Werkzeug für Werkzeugleiste
        ToolButton b = new
        ToolButton(lgcPath + "images/" +
        image,

        typname,

        new KantenTool(editor, typname),

        editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        // Eintrag in die
        // Hashtabelle
        gobjek-
        te.put(typname, kante);
        //System.out.println("In Hashta-
        //belle: " + gobjekte);

        break;
    }
    if
    (token.sval.equals("SIZE")) {
        c=token.nextToken();
        Sy-
        stem.out.println("Lese SIZE" +
        token.nval);
        break;
    }
    if
    (token.sval.equals("COLOR")) {
        //System.out.println("Lese COLOR"
        // + token.nval);
        c=token.nextToken();
        int r =
        (int)token.nval;
        c=token.nextToken();
        //System.out.println("Lese COLOR"
        // + token.nval);
        int g =
        (int)token.nval;
        c=token.nextToken();
        //System.out.println("Lese COLOR"
        // + token.nval);
        int b =
        (int)token.nval;
        color = new Co-
        lor(r,g,b);
        // Sy-
        stem.out.println("Gelesene Farbe:
        " + color);

```

```

        break;
    }
    default:
    } //switch
    c=token.nextToken();
    // Sy-
    stem.out.println("NAECHSTES
    TOKEN" + token.sval);
    } //while
    //c=token.nextToken();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // System.out.println("Bende
    readEdge");

    } //readEdge

    private void readMenu() {
        tools.clear();
        int c = '{';
        try {
            while
            ((c=token.nextToken()) != '{') {
                //c=token.nextToken();
                String namen = to-
                ken.sval;
                System.out.println("Jetzt
                kommt das Menu"+ namen);
                c = token.nextToken();
                String aufruf = to-
                ken.sval;
                System.out.println("Jetzt
                kommt das Menu"+ aufruf);
                tools.put(new
                String(namen), new
                String(aufruf));
            }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        private void readAnalyse() {
            System.out.println("Jetzt
            kommt die Analyse");
        }

        private void readShorts() {
            System.out.println("Jetzt
            kommt die Shortcut");
        }

        private void readAccel() {
            System.out.println("Jetzt
            kommen die Accelerators");
        }

```

```

// private void uebergebe
(String mpunkt,String
icon1,String icon2) {
// public void addBut-
ton(String menuePunkt, String
image1, String image2)

private void uebergebe(String
auswahl,String name,String style,
int size) {
    int styleInt = 0;
    switch (style.charAt(0)){
        case 'B':
            styleInt = Font.BOLD;
            break;
        case 'P':
            styleInt = Font.PLAIN;
            break;
        case 'I':
            styleInt = Font.ITALIC;
            break;
        default:
            styleInt = Font.PLAIN;
    }
    Font font = new Font(name,
    styleInt, size);
    switch (auswahl.charAt(0)){
        case 'M':
            edi-
            tor.getMenueleiste().setFont(font
            );
            break;
        case 'P':
            //noch zu Implementiern
            break;
        case 'S':
            edi-
            tor.getStatusleiste().setFont(fon
            t);
            break;
    }
}

private void uebergebe(String
auswahl,int r,int g,int b) {
    if (auswahl.equals("PAPER")){
        edi-
        tor.getZeichenflaeche().setBackgr
        ound(new Color(r,g,b));
    }
    if (auswahl.equals("GRID")){
        //noch zu implementiern
    }
    if
    (auswahl.equals("MENUBGC")){
        // edi-
        tor.getMenueleiste().setBackground
        d(new Color(r,g,b));
    }
    if
    (auswahl.equals("MENUFGC")){

```

```

        // menubar.setForeground(new
        Color(r,g,b));
    }
    if
    (auswahl.equals("STATUSBGC")){
        edi-
        tor.getStatusleiste().setBackgrou
        nd (new Color(r,g,b));
    }
    if
    (auswahl.equals("STATUSFGC")){
        edi-
        tor.getStatusleiste().setForegrou
        nd(new Color(r,g,b));
    }
    if
    (auswahl.equals("TOOLBGC")){
        edi-
        tor.getToolbar().setBackground(ne
        w Color(r,g,b));
    }
    if
    (auswahl.equals("TOOLFGC")){
        edi-
        tor.getToolbar().setForeground
        (new Color(r,g,b));
    }
    if
    (auswahl.equals("SHORTCUTBGC")){
        edi-
        tor.getShortcutleiste().setBackgr
        ound(new Color(r,g,b));
    }
    if
    (auswahl.equals("SHORTCUTFGC")){
        edi-
        tor.getShortcutleiste().setForegr
        ound (new Color(r,g,b));
    }
}

/**
 * Liefert eine Kopie eines
 * GraphObjektes
 * zurück.
 */
public GraphObjekt getOb-
jekt(String name) {
    if
    (gobjekte.containsKey(name)) {
        GraphObjekt vater =
        (GraphObjekt)gobjekte.get(name);
        return
        (GraphObjekt)vater.copy();
    } else {
        return null;
    }
}

/**
 * Diese Methode fügt alle an-
 * zeigbaren Objektetypen in die

```

```

 * Hashtable der Klasse Gra-
 * phObjekt ein,
 * -> alle Objekte werden ange-
 * zeigt.
 */
public void setLayer() {
    Hashtable alle = new Has-
    htable(gobjekte.size(),1.0f);
    Enumeration e = gobjek-
    te.keys();
    while (e.hasMoreElements())
    {
        String key =
        (String)e.nextElement();
        alle.put(key,new
        String(key));
    }
    GraphObjekt.toShow = alle;
}

/**
 * Liefert alle anzeigbaren
 * Layers zurück.
 */
public Enumeration getLayers()
{
    return gobjekte.keys();
}

/**
 * Liefert die maximale Anzahl
 * der Layers zurück.
 */
public int countLayers() {
    return gobjekte.size();
}

/**
 * Diese Methode fügt alle an-
 * zeigbaren AttributNamenn in die
 * Hashtable der Klasse Attri-
 * bute ein,
 * -> alle Attribute werden an-
 * gezeigt.
 */
public void setAttributNames()
{
    Hashtable alle = new Has-
    htable(attributNamen.size(),1.0f)
    ;
    Enumeration e = attributNa-
    men.keys();
    while (e.hasMoreElements())
    {
        String key =
        (String)e.nextElement();
        alle.put(key,new
        String(key));
    }
    Attribute.toShow = alle;
}

```

```

/**
 * Liefert alle anzeigbaren
AttributNamen zuruck.
 */
public Enumeration getAttributNames() {
    return attributNamen.keys();
}

/**
 * Liefert die maximale Anzahl
der Attribute zuruck.
 */
public int countAttributNamen() {
    return attributNamen.size();
}

/**
 * Fügt einen Attribut namen
in die

```

```

* Hashtabel ein.
*/
public void addAttributName (
String name) {
    attributNamen.put(new
String(name), new String(name));
}

/**
 *
 */
public Hashtable getTools() {
    return tools;
}

// public String getConfigFile()
{
    // return configFile;
    // }
}

```

## 2."load" file

```

package commands;

import etc.*;
import java.util.*;
import java.awt.*;
import java.io.*;
import interfaces.*;

/**
 * Ladet einen Graphen aus einer
.lgf Datei.
 */
public class Load extends Befehl
{
    Vector undo;

    public Load(GraphEditor edi-
tor){
        super(editor);
        undo=new Vector();
        help =
"<filename.lgf/.lgc/.lgt>";
    }

    public void ausfuehren(String[]
param){
        //System.out.println(param);
        int anzahl = param.length;
        switch (anzahl) {
            case 0 : // bei keinem Ar-
gument tun wir nichts.
                break;
            case 1 : // bei einen Ar-
gument wird erst nachgeschaut!

```

```

            if
(param[0].endsWith(".lgc") ||
        pa-
ram[0].endsWith(".lgf") ||
        pa-
85 ram[0].endsWith(".lgt") ) {
            // wir wurden
von der Commandozeile aufgerufen
            File file = new
File(param[0]);

            //System.out.println("Der Pfad :
" + file.getParent());

            //System.out.println("Der Name :
" + file.getName());

            prue-
fe(file.getParent()+"/",file.getN
ame());

            } else {
                //nothing
            }
            break;
            default : //zuviel Parame-
ter
                break;
        } //switch

    public void ausfuehren(String
param){
        edi-
tor.getStatusleiste().show("Load.
..");

        ((Component)editor).setCursor(Cur

```

```

sor.getPredefinedCursor(Cursor.WAIT_CURSOR));
FileDialog fd = new FileDialog((Frame)editor, null, FileDialog.LOAD);
// das hat leider noch keine Auswirkungen in Windows und Solaris
// ab 1.1.6 gehts doch
fd.setDirectory(System.getProperty("user.dir"));
// das schon
fd.setFile("noname.lgf");
FilenameFilter filter = new lgFilter();
fd.setFilenameFilter(filter);
fd.show();
String dir = fd.getDirectory();
String file = fd.getFile();
// fd.getFile() liefert null bei Abbruch!
if (file == null) {
    // nichts zu tun
}
((Component)editor).setCursor(Cursor.getDefaultCursor());
return;
} else {
    // laden
}
//System.out.println(fd.getDirectory());
//System.out.println(fd.getFile());
Vector gelöscht=editor.getGraph().removeAll();
pruefe(dir, file);
editor.getGraph().setChanged(false);
editor.setAuswahl(new Vector());
Vector lastCommands = editor.getLastCommands();
if (lastCommands.size() < 10) {
    lastCommands.addElement(this);
} else {
    lastCommands.removeElementAt(0);
    lastCommands.addElement(this);
}
if (undo.size() < 10) {
    undo.addElement(geloescht);
} else {
    undo.removeElementAt(0);

```

```

undo.addElement(geloescht);
}
} else {
    editor.getZeichenflaeche().drawBuffer(editor.getGraph());
}
((Component)editor).setCursor(Cursor.getDefaultCursor());
editor.getStatusleiste().show("Done");
}
/**
 * Macht Datei laden rückgängig.
 */
public void undo() {
    editor.getStatusleiste().show("Undo: Load...");
}
((Component)editor).setCursor(Cursor.getDefaultCursor(Cursor.WAIT_CURSOR));
if (!undo.isEmpty()) {
    Vector insert = (Vector)undo.lastElement();
    if (insert != null) {
        editor.getGraph().removeAll();
        editor.getGraph().add(insert);
        insert.removeAllElements();
    }
}
undo.removeElement(undo.lastElement());
}
editor.getZeichenflaeche().drawBuffer(editor.getGraph());
editor.getGraph().setChanged(true);
editor.getStatusleiste().show("Done");
}
((Component)editor).setCursor(Cursor.getDefaultCursor());
} //undo
/**
 * Wiederholt Datei laden..
 */
public void redo() {
    editor.getStatusleiste().show("Redo: Load...");
    ausfuehren();
}

```

```

    } // redo

    /**
     * Diese Klasse wird leider
     nicht an
     * die Windows bzw Solaris Kom-
     ponente
     * weitergereicht.
     */
    class lgFilter implements Fi-
    lenameFilter {
        public boolean accept (File
        dir, String name) {
            return ( na-
            me.endsWith(".lgf") ||
            na-
            me.endsWith(".lgc") ||
            na-
            me.endsWith(".lgt") );
        }
    }

    /**
     * Diese Methode überprüft, ob
     die richtige
     * Konfigurationsdatei geladen
     ist, ansonsten wird
     * versucht die richtige zu la-
     den. (->Editor zurücksetzen)
     * Dannach wird die gewünschte
     .lgt oder .lgf Datei
     * geladen.
     */
    private void pruefe (String
    pfad, String datei) {
        Einstellungen settings= edi-
        tor.getEinstellungen();
        if (datei.endsWith(".lgc")) {
            //System.out.println("eine
            lgc Datei");
            File f = new File(pfad +
            datei);
            if (f.exists()) {
                settings.appName = Ein-
                stellungen.format(datei);
                settings.fileName=" ";
                settings.frameName = set-
                tings.fileName+ " "
                +settings.appName + " "
                +settings.copyright;
                settings.configFile = new
                String(datei);
                settings.lgcPath = new
                String(pfad);
                //wir Starten den Editor
                neu
                editor.start();
            } else {
                System.err.println("File
                not found : "+ settings.lgcPath +
                datei);
            }
        }
    }

```

```

    } else if
    (datei.endsWith(".lgf")) {
        //System.out.println("eine
        lgf Datei");
        File f = new File(pfad +
        datei);
        if (f.exists()) {
            settings.fileName = da-
            tei;
            // wir holen uns noch den
            namen des .lgc Files:
            String config = edi-
            tor.getDateischnittstelle().getCo-
            nfig(pfad + datei);
            //System.out.println("Der
            neue Name der Lgc datei " + con-
            fig);
            f = new
            File(settings.lgcPath + config);
            if (f.exists()) {
                // ist diese lgc Datei
                schon geladen?
                if
                (settings.configFile.equals(confi-
                g)) {
                    //wir muessen nur die
                    lgf Datei laden
                    edi-
                    tor.getDateischnittstelle().load(
                    pfad,datei,editor.getGraph());
                    settings.frameName =
                    settings.fileName+ " "
                    +settings.appName + " "
                    +settings.copyright;
                    ((Frame)editor). set-
                    Title(settings.frameName);
                } else {
                    // wir müssen auch
                    die Konfigurationsdatei laden
                    settings.appName =
                    Einstellungen.format(config);
                    settings.configFile =
                    new String(config);
                    settings.frameName =
                    settings.fileName+ " "
                    +settings.appName + " "
                    +settings.copyright;
                    //wir Starten den
                    Editor neu
                    editor.start();
                    edi-
                    tor.getDateischnittstelle().load(
                    pfad,datei,editor.getGraph());
                }
            } else {
                Sy-
                stem.err.println("File not found
                : " + settings.lgcPath + config);
            }
        } else {
            System.err.println("File
            not found : " + pfad + datei);
        }
    }

```

```

    }
    //start();
    } else if
    (datei.endsWith(".lgt")) {
        //System.out.println("eine
        lgt Datei");
        File f = new File(pfad +
        datei);
        if (f.exists()) {
            settings.fileName = da-
            tei;
            settings.frameName = set-
            tings.fileName+ " "
            +settings.appName + " "
            +settings.copyright;
            // wir holen uns noch den
            namen des .lgc Files:
            //String config = edi-
            tor.getDateischnittstelle().getCo-
            nfig(pfad + datei);
            //System.out.println("Der
            neue Name der Lgc datei " + con-
            fig);
            //f = new
            File(settings.lgcPath + config);
            //if (f.exists()) {
            // ist diese lgc Datei
            schon geladen?
            //if
            (settings.configFile.equals(confi-
            g)) {
                //wir muessen nur die
                lgt Datei laden und interpretie-
                ren
                LgtInterpreter inter-
                preter=editor.getInterpreter();

                //System.out.println("Der Inter-
                preter : " + interpreter);
                if (interpreter ==
                null) {
                    interpreter = new
                    LgtInterpreter(editor, pfad + da-
                    tei);
                    edi-
                    tor.setInterpreter(interpreter);
                    interpreter-
                    ter.start();
                } else {

```

```

                    interpre-
                    ter.setFile(pfad + datei);
                }

                //Dateischnittstelle().load(pfad,
                datei, editor.getGraph());
                //settings.frameName
                = settings.appName + " " + set-
                tings.fileName;
                //((Frame)editor).
                setTitle(settings.frameName);
                // } else {
                // wir muessen auch
                die Konfigurationsdatei laden
                // settings.appName =
                Einstellungen.format(config);
                //settings.configFile
                = new String(config);
                //settings.frameName
                = settings.appName + " " + set-
                tings.fileName;
                //wir Starten den
                Editor neu
                //editor.start();
                // LgtInterpreter in-
                terpreter = new LgtInterpre-
                ter(editor, pfad + datei);
                // edi-
                tor.setInterpreter(interpreter);
                // interpre-
                ter.start();
                // }
                //} else {
                // Sy-
                stem.err.println("File not found
                : " + settings.lgcPath + config);
                //}
                } else {
                    System.err.println("File
                    not found : " + pfad + datei);
                }
                } else {
                    System.err.println("usage:
                    java LoGraph2 <path to config-
                    files> AND <file.lgc> OR
                    <file.lgf> OR <file.lgt>");
                }
            }
        }
    }
}

```

### 3. "toolbar" file

```

package mm1;

import java.awt.*;
import java.awt.event.*;

import etc.*;
import tools.*;

```

```

/**
 * Über das aktuelle Tool der
 * Toolbar werden die
 * Maus Aktionen des Benutzers an
 * den Graphen weitergegeben.

```



```

* Die Toolbar ermöglicht das
hinzufügen und entfernen
* von ToolButtons, und deren zu-
gehörigen ActionListener.
*/
public class Toolbar extends Pa-
nel {
    GraphEditor editor;
    Tool currentTool;
    ToolButton currentButton;
    int borderSize = 4;
    /**
     * Der Konstruktor erzeugt das
     AuswahlTool,
     * da dieses immer vorhanden
     sein sollte.
     */
    public Toolbar(GraphEditor edi-
    tor) {
        this.editor = editor;
        setLayout(new BarLay-
        out(BarLayout.VERTICAL, 2));
        setBackground-
        und(editor.getEinstellungen().too-
        lbarBgCo);
        // eine kleine Lücke
        add(new Space(5, 24));

        ToolButton b = new ToolBut-
        ton(editor.getEinstellungen().lge-
        Path +

        "images/auswahl.gif",

        "Select",

        new AuswahlTool(editor), this);
        setCurrentTool(b.getTool());
        setCurrentButton(b);
        add(b);
        add(new Space(5, 24));
    }

    public Insets getInsets() {
        Insets insets =
        (Insets) (super.getInsets()).clone
        ();
        insets.top += borderSize;
        insets.left +=
        (borderSize+2);
        insets.bottom += borderSize;
        insets.right +=
        (borderSize+2);
        return insets;
    }

    public void paint(Graphics g) {
        super.paint(g);
        Insets insets = su-
        per.getInsets();

```

```

        int w = getSize().width-
        insets.left-insets.right;
        int h = getSize().height-
        insets.top-insets.bottom;

        g.setColor(editor.getEinstellunge-
        n().toolbarBgCo);
        for (int i=0; i<borderSize;
        i++) {

            g.draw3DRect(i+insets.left,i+inse-
            ts.top,

                                w-2*i-1, h-
            2*i-1, i<borderSize/2);
        }

        /**
         * Fügt einen ToolButton hinzu.
         */
        public void addToolBut-
        ton(ToolButton button) {
            add(button);
        }

        /**
         * Entfernt einen ToolButton.
         */
        public void deleteTooleBut-
        ton(ToolButton button) {
        }

        /**
         * Setzt das aktuelle Tool;
         * wird normalerweise von den
         ToolButtons aufgerufen.
         */
        public void setCurrentTool(Tool
        currentTool) {
            this.currentTool = current-
            Tool;
            this.currentTool.reset();
        }

        /**
         * Setzt den aktuellen Button,
         damit der nächste
         * aktuelle Button ihn zurück-
         setzen kann.
         */
        public void setCurrentBut-
        ton(ToolButton currentButton) {
            if (this.currentButton !=
            null)
                this.currentButton.setUp();
            this.currentButton = current-
            Button;
            this.currentButton.setDown();
        }

        /**

```

```

    * Liefert das aktuelle Tool
    zurück.
    * wird normalerweise von den
    Zeichenfläche aufgerufen.
    */
    public Tool getCurrentTool() {
        return currentTool;
    }

    /**
    * Liefert den aktuellen But-
    ton, damit der nächste
    * aktuelle Butten ihn zurück-
    setzen kann.
    */

```

```

    public ToolButton getCurrent-
    Button() {
        return currentButton;
    }

    /**
    * Liefert den Editor an die
    Buttons weiter.
    */
    public GraphEditor getEditor()
    {
        return editor;
    }

} //Toolbar

```

This redlined draft, generated by CompareRite (TM) - The Instant Redliner, shows the differences between -

original document : Q:\DOCUMENTS\YEAR 2001\P010041-THURNER-  
DETERMINING A GRAPHIC STRUCTURE\ORIGINAL SPECIFICATION.DOC

5 and revised document: Q:\DOCUMENTS\YEAR 2001\P010041-THURNER-  
DETERMINING A GRAPHIC STRUCTURE\SUBSTITUTE SPECIFICATION.DOC

CompareRite found 183 change(s) in the text

10 Deletions appear as Overstrike text surrounded by []  
Additions appear as Bold-Underline text

[Description] **SPECIFICATION**

[Method for determining a graphic structure of a technical system and arrangement

15 and set of arrangements for determining] **TITLE**

**METHOD FOR DETERMINING A GRAPHIC STRUCTURE OF A TECHNICAL**

**SYSTEM AND ARRANGEMENT AND SET OF ARRANGEMENTS FOR**

**DETERMINING A GRAPHIC STRUCTURE**

**BACKGROUND OF THE INVENTION**

20 **Field of the Invention**

**1 The invention relates to the selection of elements of a graph structure file  
in order to describe the structure of a techical system graphically.**

**Description of the Related Art**

25 **2 It is known to describe different technical systems by means of a graphic  
structure. Such descriptions are known from, for example, product brochures  
for products provided by Zuken-Redac (e.g., Analysis Products, CAD Products,  
CAE Products, CAM Products, and Data Conversion Products—formerly  
available on September 22, 1998 at**

30 **http://www.redac.co.uk/prod info/brochures/14a.html) (the Zuken-Redac  
brochures), herein incorporated by reference, that disclose[[1]-discloses] how,  
for a technical system such as an electronic circuit, the electrical circuit is determined  
in the form of a graphic structure with elements which describe an electronic circuit.**

**3 Elements of a [graph] graphic structure in the field of a circuit simulation are**

symbols which symbolize electronic components, for example, a resistor, a capacitor, an inductor, a transistor, an operational amplifier or other electronic components composed of these elements.

4 In the method ~~[known from [1]]~~ and ~~[the]~~ arrangement known from ~~[[1]]~~ the  
5 Zuken-Redac brochures, elements for graphically describing an electronic circuit which are made available to a user by an editor program are selected in such a way that the "electronic circuit" constituting the technical system is described using the selected elements. The elements are represented by the editor program.

5 A ~~[graph]~~ graphic structure describes a ~~[graph]~~ graphic  $G (= V, E, \Psi)$  which  
10 has a finite, non-empty set  $V$  ( $v \in \blacksquare V$  designate nodes of the ~~[graph]~~ graphic  $G$ ), and a finite set  $E$  ( $e \in \blacksquare E$  designate edges of the ~~[graph]~~ graphic  $G$ ). The nodes and edges of the ~~[graph]~~ graphic are logically combined by an incidence function  $\Psi$  which is formed according to the following rule:

$$\Psi: E \rightarrow \{\{i, j\} \mid i, j \in \blacksquare V\} \quad (1)$$

15 6 Each edge  $e$  of the set  $E$  of edges is assigned its two end places by the incidence function  $\Psi(e)$ .

7 Depending on the field of application, different types of nodes and edges may be provided in an editor program for describing a technical system. Nodes and edges to which an application-dependent semantic is assigned are generally  
20 designated as elements of the ~~[graph]~~ graphic in an editor program. A node element of a ~~[graph]~~ graphic is, for example in the editor program in ~~[[1]]~~ the Zuken-Redac brochures, a symbol which symbolizes an electronic component of the electronic circuit. The edges can be used to describe weighted connections between the individual elements. Generally, the respective nodes and edges can be assigned  
25 a weight, a value or any desired text for information (textual information).

~~[[2]]~~ 8 G. Chiola, G. Franceschinis, R. Gaeta and M. Ribaud, GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets, Performance Evaluation, special issue on Performance Modeling Tools, 24 (1&2), pp. 47 - 68, November 1995 (Chiola), herein incorporated by reference,

30 discloses an editor program for determining a Petri net. A Petri net is preferably used to analyze and design a closed-loop control system or an open-loop control system of a technical system, generally for describing system characteristics of a technical

system. A ~~[graph]~~ **graphic**, which is illustrated in the form of a Petri net, has a place S or a transition T as elements. A general overview of a Petri net and its basic elements can be found in ~~[[3]]~~.

**G. Schmidt, Grundlagen der Regelungstechnik: Analyse und Entwurf linearer und einfacher nichtlinearer Regelungen sowie diskreter Steuerungen**  
**[Principles of control technology: analysis and design of linear and simple nonlinear closed-loop controls and discrete open-loop controls], second edition, Springer-Verlag [Publishing House], ISBN 3-540-17112-6, Berlin, pp. 320 - 328, 1991 (Schmidt), herein incorporated by reference.**

10 **9** A Petri net is generally a triplet

$$N := \langle S, T, F \rangle$$

where

(i)  $S = \{ s_1, s_2, \dots, s_n \}$

Set of places

(ii)  $T = \{ t_1, t_2, \dots, t_m \}$

Set of transitions

15 (iii)  $S \cap T = \emptyset$

S and T disjunctive

(the node set is

composed of S and T)

(iv)  $F \subseteq (S \times T) \cup (T \times S)$

Flow relation

20 **10** A **particular** disadvantage with the known methods and arrangements is ~~fin~~  
~~particular]~~ the fact that in each case elements of a ~~[graph]~~ **graphic** which are  
provided only for a specific application are made available as a function of the  
application in order to determine the graphic structure of a technical system. Thus,  
with the editor program from ~~[[1]]~~ **the Zuken-Redac brochures**, only a selection of  
25 the elements can be made to describe an electronic circuit, and in the case of the  
editor program from ~~[[2]]~~ **Chiola**, only a selection of elements can be made to  
describe a Petri net.

**11** Such a known editor program is thus extremely inflexible in a situation in  
which a user wishes to use different types of a graphic structure to describe a  
30 technical system. ~~[It is then]~~ **In this type of program, it is** necessary to develop for  
each specific application a separate editor program which is adapted to the  
application, something which entails considerable development costs.

## SUMMARY OF THE INVENTION

12 The invention is therefore based on ~~[the problem of disclosing]~~ providing a method for determining a graphic structure of a technical system, and an arrangement and a set of a plurality of arrangements for determining a ~~[graph]~~ graphic structure which has improved flexibility in comparison with the known methods and arrangements.

13 The problem is solved by a ~~[means of the method, the arrangement and the set of arrangements according to the features of the independent patent claims.~~  
A) method for determining a graphic structure of a technical system (which may be  
10 an electronic circuit or a piece of technical equipment) has the following steps:

14 a) a ~~[graph]~~ graphic structure file is selected from a set of a plurality of different ~~[graph]~~ graphic structure files, a ~~[graph]~~ graphic structure file containing, in each case, indications of which elements can be selected to represent ~~[it]~~ the graphic structure file in order to describe the structure of the technical system graphically,

15 15 b) elements are selected in such a way that a technical system is described using the selected elements, and

16 c) the elements are represented by an editor program into which the selected ~~[graph]~~ graphic structure file has been integrated, ~~[by]~~ via which ~~[means]~~ the graphic structure of the technical system is determined.

20 ~~[An]~~ 17 The problem is also solved by an arrangement for determining a ~~[graph]~~ graphic structure has the following features:

18 a) a memory in which a set of a plurality of different ~~[graph]~~ graphic structure files are stored, a ~~[graph]~~ graphic structure file containing, in each case, indications of which elements can be selected to represent it in order to form a ~~[graph]~~ graphic,

25 19 b) a selector unit with which a ~~[graph]~~ graphic structure file can be selected from the set of ~~[graph]~~ graphic structure files,

20 c) a processor ~~[which is]~~ configured ~~[in such a way that]~~ to execute an editor program ~~[can be executed]~~, with which editor program a ~~[graph]~~ graphic structure file selected from the set of ~~[graph]~~ graphic structure files can be used to determine a ~~[graph]~~ graphic with elements of the selected ~~[graph]~~ graphic structure file, by which means the ~~[graph]~~ graphic structure is determined, and

30 21 d) a representation component which is coupled to the editor program and

with which the specific [graph] **graphic** structure can be represented.

**22** In the inventive method, the elements may be graphic elements of a graphic which describes the technical system. Also, a further step of checking the graphic structure of the technical system for predefined structure rules may be provided as well.

**23** A set of a plurality of arrangements for determining a [graph] **graphic** structure has:

**24** a) a first arrangement which has a memory in which a set of a plurality of different [graph] **graphic** structure files are stored, a [graph] **graphic** structure file containing in each case indications of which elements can be selected to represent it in order to form a [graph] **graphic**, and

**25** b) a second arrangement which is coupled to the first arrangement and has the following components:

**26** - a selector unit with which a [graph] **graphic** structure file can be selected from the set of [graph] **graphic** structure files,

**27** - an editor program with which a [graph] **graphic** structure file selected from the set of [graph] **graphic** structure files can be used to determine a [graph] **graphic** with elements, of the selected [graph] **graphic** structure file, [by] **via** which [means] the [graph] **graphic** structure is determined, **and**

**28** - a representation component which is coupled to the editor program and with which the specific [graph] **graphic** structure can be represented.

**29** The invention discloses a method which is very flexible in comparison with the known methods and arrangements, and a very flexible arrangement for determining a graphic structure which can be adapted to new application scenarios in a quick and [uncomplicated] **easy** way[,] and can be adapted more satisfactorily to existing application scenarios.

**30** In this way, different types of structures which can be represented as a [graph] **graphic** can be processed flexibly, cost-effectively, and easily with [a] **the inventive** method or [with an] arrangement.

~~[Preferred developments of the invention emerge from the dependent claims.]~~

**31** These inventive aspects are described in more detail below.

**32** The technical system is preferably an electronic circuit or a piece of technical equipment. The elements are preferably [graph] **graphic** elements of a [graph]

**graphic** which describe the technical system.

**33** In a further refinement there is provision for the graphic structure of the technical system which is determined to be checked for predefined structure rules. In this way, it is possible to check a structure of the technical system determined by a user for predefined structure rules, which ensures that the structure rules for the respective technical system are complied ~~[with in]~~ **within** terms of its graphic structure.

~~[Such a]~~ **34** **An exemplary** structure rule ~~[is]~~ **could be**, for example, in a Petri net, the fact that a place must always follow a transition, and vice versa. If this is not the case, within the scope of this development, the disclosure is made during checking of the graphic structure of a Petri net that the corresponding structure rule is infringed.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

**35** An exemplary embodiment of the invention is illustrated in the figures and explained in more detail below.

~~[In said figures:~~

~~]~~**36** Figure 1 ~~[shows an outline of]~~ **is a schematic diagram showing** an arrangement according to a first exemplary embodiment;

**37** Figure 2 ~~[shows an outline]~~ **is a pictorial diagram** of a representation component with a graphic structure with elements of a Petri net;

**38** Figure 3 ~~[shows an outline]~~ **is a pictorial diagram** of a representation component with a graphic structure with elements which describe an electronic circuit;

**39** Figure 4 ~~[shows]~~ **is** a flowchart in which the method steps of the method according to an exemplary embodiment are represented; **and**

**40** Figure 5 ~~[shows]~~ **is a block diagram of** a set of a plurality of arrangements which, according to a second exemplary embodiment, are coupled to one another ~~[by means of]~~ **via** a communications network.

### **DESCRIPTION OF THE PREFERRED EMBODIMENTS**

**41** Fig. 1 shows an arrangement 100 with a set 101 of a plurality of different graphic structure files 102, 103, 104, 105. Each graphic structure file 102, 103, 104,



105 is embodied as a file which can be linked dynamically (dynamic link library).

**42** A user 106 selects a graphic structure file 102, 103, 104, 105 using a selection component 108 (keyboard and/or computer mouse) which is connected to an editor program 107.

5 **43** The selected graphic structure file, in this exemplary embodiment a first graphic structure file 103, is dynamically integrated into the editor program 107.

**44** After integration into the editor program 107, a set 111 of selectable elements 112, 113, 114, which are defined in the first graphic structure file 103 as selectable elements for determining a further described ~~[graph]~~ **graphic** is displayed to the user  
10 106 on a screen 110 ~~[by means of]~~ via a representation component 109 which is connected to the editor program 107. In addition, in this exemplary embodiment, according to the first graphic structure file 103, a first check program 115 and a second check program 116 are integrated into the editor program 107 and made available to the user 106 for selection.

15 **45** Each graphic structure file 102, 103, 104, 105 has, in each case, a set of selectable elements for the respective type of ~~[graphs]~~ **graphics**, in each case a graphic structure file being provided for one type of ~~[graph]~~ **graphic**. In addition, each graphic structure file 102, 103, 104, 105 may respectively contain a specific check program which is integrated into the respective graphic structure file 102, 103,  
20 104, 105.

**46** After the first graphic structure file 103 has been integrated, elements of the ~~[graph]~~ **graphic** are selected by the user 106 and connected to one another so that a ~~[graph]~~ **graphic** is determined which is stored in the form of a predefined intermediate language 117 in a memory 118.

25 **47** In addition, ~~[it is symbolically represented in Fig. 1]~~ **Fig. 1 symbolically represents** that the user 106 stores a plurality of structures 119, 120, 121, 122, 123 for describing different ~~[graphs, said]~~ **graphics, these** structures relating to the type of ~~[graph]~~ **graphic** predefined by the first graphic file 103. The first graphic structure file 103 makes available elements which make possible a ~~[graph]~~ **graphic** in the  
30 form of a Petri net 201 (see Fig. 2).

**48** Fig. 2 shows the representation component 200 which is presented to the user 106 in the form of a screen surface. The screen surface 200 has a menu list 202 with different selectable options ("File", "Edit", "Object", "View", "Tools",

"Settings", "Help"). Menu items are made available to the user by means of individual selectable elements using an immediate access bar 203 by making a single, direct selection of an element.

**49** In addition, a processing bar 204 is represented with selectable options for determining the ~~[graph]~~ **graphic**. Thus, in the first graphic structure file 103, a first selection element 205 is provided with which it is possible to select and process objects represented on the screen. The selection and processing of specific elements for a Petri net 201 is made available to the user 106 ~~[by means of]~~ via a set 206 of further selector elements which will be explained in more detail below.

**50** A second selector element 207 is described by ~~[means of]~~ an empty rectangle and symbolizes a time-specific transition. A third selector element 208 symbolizes a timeless transition, which is represented as a selected transition element 220, 221 and 222 in the Petri net 201. A fourth selector element 209 symbolizes an edge which is a directed edge in this exemplary embodiment. A fifth selector element 210 symbolizes a forbidden edge which is designated in accordance with the structure rules of a Petri net 201. A sixth selector element 211 symbolizes a place where, in each case, a place element 223, 224, 225, 226 ~~[being]~~ is represented in the Petri net 201. The place elements 223, 224, 225 and 226 are connected to the transition elements 220, 221, 222 via edges 227, 228, 229, 230, 231 and 232. A seventh selector element 212 symbolizes the possibility of combining a plurality of elements of the Petri net to form a composite element. An eighth selector element 213 symbolizes an input of the Petri net 201 and a ninth selector element 214 symbolizes an output of a Petri net 201.

**51** The edges and the individual nodes, ~~[that is to say]~~ i.e., the elements of the Petri net 201, are assigned textual information 251, 252, 253, 254, 255, 256, 257, 258, 259, 260 and 261. In this way it is possible to assign an additional textual description to the individual elements.

**52** If a second graphic structure file 104 is integrated into the editor program 107, the second graphic structure file 104 making available elements of an electronic circuit, and thus a ~~[graph]~~ **graphic** of an electronic circuit, a screen mask represented in Fig. 3 with a set of selector elements set up for the circuit simulation is produced.

**53** The same designations are used in Fig. 3 for the same elements displayed on

the screen as represented in Fig. 2.

**54** A set 301 of selector elements which are specifically for describing a ~~[graph]~~  
graphic of an electronic circuit contain :

- a tenth selector element 302 which symbolizes an electronic resistor,
- 5       • an eleventh selector element 303 which symbolizes an electronic capacitor,
- a twelfth selector element 304 which symbolizes an inductor,
- a thirteenth selector element 305 symbolizing a transistor,
- a fourteenth selector element 306 symbolizing an operational
- 10   amplifier,
- a fifteenth selector element 307 symbolizing a non-directed edge, and
- a sixteenth selector element 308 symbolizing a power source.

**55** An electronic circuit 110 is determined by the user 106 and has, in this exemplary embodiment, a power source 311, electronic resistors 312, 313,  
15   electronic capacitors 314 and 315 and an operational amplifier 316 which are each connected to one another ~~[by means of]~~ via edges 317. In addition, a ground terminal 318 is illustrated in Fig. 3. The individual circuit elements are assigned textual information 319, 320, 321, 322, 323, 324, 325, 326 for further explaining the electronic circuit 310.

**56** Fig. 4 shows the ~~[method in its method steps in order to clarify the method]~~  
20   inventive method steps. In a first step (step 401) a graphic structure file 102, 103, 104, 105 is selected from a set 101 of graphic structure files 102, 103, 104, 105. In a second step (step 402), a selection is made of elements which are available in accordance with the graphic structure file 102, 103, 104, 105 which was selected in the previous step (step 401). The selected elements are illustrated by the editor program 107 in a further step (step 403).

**57** Fig. 5 shows a first computer 500 with a memory 502 and a processor 503 which are each connected to one another ~~[by means of]~~ via a bus 504 and to an input/output interface 501. The first computer 500 is connected to a screen 505, a  
30   keyboard 506, and a computer mouse 507 ~~[by means of]~~ via the input/output interface 501.

**58** In addition, the first computer 500 is connected to further computers 510, 520,

530, 540 and 550 via a communications network 560, in the exemplary embodiment, an ISDN network (Integrated Services Digital Network).

~~[The set 101 of graphic structure files 102, 103, 104, 105 is stored in the first computer 500.]~~

5 **59** **The set 101 of graphic structure files 102, 103, 104, 105 is stored in the first computer 500.** The further computers 510, 520, 530, 540 and 550 each also have a processor 513, 523, 533, 543 and 553 and each have a memory 512, 522, 532, 542 and 552. In each case the processor 513, 523, 533, 543 and 553 and the memory 512, 522, 532, 542 and 552 are connected to the communications network  
10 via, in each case, a bus 514, 524, 534, 544 and 554 via an input/output interface 511, 521, 531, 541 and 551. In addition, the further computers 510, 520, 530, 540 and 550 are each connected to a screen 515, 525, 535, 545 and 555, to a keyboard 516, 526, 536, 546 and 556 and to a computer mouse 517, 527, 537, 547 and 557.

**60** An editor program 508, 518, 528, 538, 548, 558 is stored in each computer  
15 500, 510, 520, 530, 540 and 550. A graphic structure file 102, 103, 104, 105 is selected by a user of a further computer 510, 520, 530, 540 and 550, and requested from the first computer 500 with a request message 570. The first computer 500 transmits the selected graphic structure file 102, 103, 104, 105 in a reply message 580 to the further computer 510, 520, 530, 540 and 550 requesting the graphic  
20 structure file 102, 103, 104, 105.

**61** The requesting further computer 510, 520, 530, 540 and 550 has thus received the requested graphic structure file 102, 103, 104, 105, and it integrates it into its editor program 518, 528, 538, 548, 558, as described in the first exemplary embodiment.

25 **62** A number of alternatives to the exemplary embodiments described above are illustrated ~~[below]~~ **as follows**: The type of elements which are made available by a graphic structure file is generally freely selectable and depends only on the respective type of ~~[graph]~~ **graphic** to be determined. The technical system can, for example, also be a piece of technical equipment whose characteristics or structure  
30 can be described by the ~~[graph]~~ **graphic**. The editor program and the ~~[graph]~~ **graphic** illustrated with the editor program can be used as part of a simulation of the technical system.

~~[The following publications are cited in this document:]~~

[1] ~~Publication available on the Internet on September 2, 1998 at the address:~~

~~http://www.redac.co.uk/prod\_info/brochures/14a.html~~

[2] ~~G. Chiola, G. Franceschinis, R. Gaeta and M. Ribaudo, GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets, Performance Evaluation,~~

5 ~~special issue on Performance Modeling Tools, 24 (1&2), pp. 47 - 68, November 1995~~

[3] ~~G. Schmidt, Grundlagen der Regelungstechnik: Analyse und Entwurf linearer und einfacher nichtlinearer Regelungen sowie diskreter Steuerungen [Principles of control technology: analysis and design of linear and simple nonlinear closed-loop controls and discrete open-loop controls], second edition, Springer-Verlag~~

10 ~~[Publishing House], ISBN 3-540-17112-6, Berlin, pp. 320 - 328, 1991~~

~~A way of implementing the exemplary described above is given below,} 63 **Three files are provided in the Appendix which implement the exemplary**~~

~~**embodiments** written in the **C/Java** programming language [C, the implementation being divided into three files:]. **These files are: 1) an initialization file, 2) a load**~~

15 ~~**file, and 3) a toolbar file.**~~

~~[1. Initialization file:~~

~~Abstract~~

~~Method for determining a graphic structure of a technical system] 64      **The**~~  
20 ~~**above-described method** and arrangement [and set of arrangements for determining a graph structure] **are illustrative of the principles of the present invention. Numerous modifications and adaptations will be readily apparent to those skilled in this art without departing from the spirit and scope of the present invention.**~~

25

[A-graph] **ABSTRACT**

- 65**     **A graphic** structure file is selected from a set of a plurality of different [graph] **graphic** structure files. A [graph] **graphic** structure file contains in each case
- 5     indications of which elements can be selected to represent it in order to describe the structure of the technical system graphically. Elements are selected in such a way that the selected elements describe the technical system, and the elements are represented by an editor program into which the selected [graph] **graphic** structure file has been integrated.

**Description**

Method for determining a graphic structure of a  
technical system and arrangement and set of  
5 arrangements for determining a graph structure

It is known to describe different technical systems by means of a graphic structure.

[1] discloses how, for a technical system such  
10 as an electronic circuit, the electrical circuit is determined in the form of a graphic structure with elements which describe an electronic circuit.

Elements of a graph structure in the field of a circuit simulation are symbols which symbolize  
15 electronic components, for example a resistor, a capacitor, an inductor, a transistor, an operational amplifier or other electronic components composed of these elements.

In the method known from [1] and the  
20 arrangement known from [1], elements for graphically describing an electronic circuit which are made available to a user by an editor program are selected in such a way that the "electronic circuit" constituting the technical system is described using  
25 the selected elements. The elements are represented by the editor program.

A graph structure describes a graph  $G (= V, E, \Psi)$  which has a finite, non-empty set  $V$  ( $v \in V$  designate nodes of the graph  $G$ ), and a finite set  $E$  ( $e \in E$  designate edges of the graph  $G$ ). The nodes and  
30 edges of the graph are logically combined by an incidence function  $\Psi$  which is formed according to the following rule:

$$\Psi: E \rightarrow \{\{i, j\} \mid i, j \in V\} \quad (1)$$

35

Each edge  $e$  of the set  $E$  of edges is assigned its two end places by the incidence function  $\Psi(e)$ .

Depending on the field of application, different types of nodes and edges may be provided in an editor program for describing a technical system. Nodes and edges to which an application-dependent semantic is assigned are generally designated as elements of the graph in an editor program.

A node element of a graph is, for example in the editor program in [1], a symbol which symbolizes an electronic component of the electronic circuit. The edges can be used to describe weighted connections between the individual elements.

Generally, the respective nodes and edges can be assigned a weight, a value or any desired text for information (textual information).

[2] discloses an editor program for determining a Petri net. A Petri net is preferably used to analyze and design a closed-loop control system or an open-loop control system of a technical system, generally for describing system characteristics of a technical system. A graph, which is illustrated in the form of a Petri net, has a place  $S$  or a transition  $T$  as elements. A general overview of a Petri net and its basic elements can be found in [3].

A Petri net is generally a triplet  
$$N := \langle S, T, F \rangle$$

where



- (i)  $S = \{ s_1, s_2, \dots, s_n \}$  Set of places
- (ii)  $T = \{ t_1, t_2, \dots, t_m \}$  Set of transitions
- (iii)  $S \cap T = \emptyset$  S and T disjunctive  
(the node set is  
composed of S and T)
- (iv)  $F \subseteq (S \times T) \cup (T \times S)$  Flow relation

5  
10 A disadvantage with the known methods and arrangements is in particular the fact that in each case elements of a graph which are provided only for a specific application are made available as a function of the application in order to determine the graphic structure of a technical system. Thus, with the editor program from [1], only a selection of the elements can  
15 be made to describe an electronic circuit, and in the case of the editor program from [2] only a selection of elements can be made to describe a Petri net.

Such a known editor program is thus extremely  
inflexible in a situation in which a user wishes to use  
20 different types of a graphic structure to describe a technical system. It is then necessary to develop for each specific application a separate editor program which is adapted to the application, something which entails considerable development costs.

25 The invention is therefore based on the problem of disclosing a method for determining a graphic structure of a technical system, and an arrangement and a set of a plurality of arrangements for determining a graph structure which has improved flexibility in  
30 comparison with the known methods and arrangements.

The problem is solved by means of the method, the arrangement and the set of arrangements according to the features of the independent patent claims.

A method for determining a graphic structure of a technical system has the following steps:

a) a graph structure file is selected from a set of a plurality of different graph structure files,  
5 a graph structure file containing in each case indications of which elements can be selected to represent it in order to describe the structure of the technical system graphically,

b) elements are selected in such a way that a  
10 technical system is described using the selected elements, and

c) the elements are represented by an editor program into which the selected graph structure file has been integrated, by which means the graphic  
15 structure of the technical system is determined.

An arrangement for determining a graph structure has the following features:

a) a memory in which a set of a plurality of different graph structure files are stored, a graph  
20 structure file containing in each case indications of which elements can be selected to represent it in order to form a graph,

b) a selector unit with which a graph structure file can be selected from the set of graph structure  
25 files,

c) a processor which is configured in such a way that an editor program can be executed, with which editor program a graph structure file selected from the set of graph structure files can be used to determine a  
30 graph with elements of the selected graph structure file, by which means the graph structure is determined, and

d) a representation component which is coupled to the editor program and with which the specific graph  
35 structure can be represented.

A set of a plurality of arrangements for determining a graph structure has:

a) a first arrangement which has a memory in which a set of a plurality of different graph structure files are stored, a graph structure file containing in each case indications of which elements can be selected to represent it in order to form a graph, and

b) a second arrangement which is coupled to the first arrangement and has the following components:

- a selector unit with which a graph structure file can be selected from the set of graph structure files,

- an editor program with which a graph structure file selected from the set of graph structure files can be used to determine a graph with elements, of the selected graph structure file, by which means the graph structure is determined,

- a representation component which is coupled to the editor program and with which the specific graph structure can be represented.

The invention discloses a method which is very flexible in comparison with the known methods and arrangements, and a very flexible arrangement for determining a graphic structure which can be adapted to new application scenarios in a quick and uncomplicated way, and can be adapted more satisfactorily to existing application scenarios.

In this way, different types of structures which can be represented as a graph can be processed flexibly, cost-effectively and easily with a method or with an arrangement.

Preferred developments of the invention emerge from the dependent claims.

The technical system is preferably an electronic circuit or a piece of technical equipment.

The elements are preferably graph elements of a graph which describe the technical system.

In a further refinement there is provision for the graphic structure of the technical system which is  
5 determined to be checked for predefined structure rules. In this way, it is possible to check a structure of the technical system determined by a user for predefined structure rules, which ensures that the structure rules for the respective technical system are  
10 complied with in terms of its graphic structure.

Such a structure rule is, for example, in a Petri net, the fact that a place must always follow a transition, and vice versa. If this is not the case, within the scope of this development the disclosure is  
15 made during checking of the graphic structure of a Petri net that the corresponding structure rule is infringed.

An exemplary embodiment of the invention is illustrated in the figures and explained in more detail  
20 below. In said figures:

Figure 1 shows an outline of an arrangement according to a first exemplary embodiment;

Figure 2 shows an outline of a representation component with a graphic structure with elements of a  
25 Petri net;

Figure 3 shows an outline of a representation component with a graphic structure with elements which describe an electronic circuit;

Figure 4 shows a flowchart in which the method steps of the method according to an exemplary embodiment are represented;

5 Figure 5 shows a set of a plurality of arrangements which, according to a second exemplary embodiment, are coupled to one another by means of a communications network.

Fig. 1 shows an arrangement 100 with a set 101 of a plurality of different graphic structure files 102, 103, 104, 105. Each graphic structure file 102, 103, 104, 105 is embodied as a file which can be linked dynamically (dynamic link library).

15 A user 106 selects a graphic structure file 102, 103, 104, 105 using a selection component 108 (keyboard and/or computer mouse) which is connected to an editor program 107.

The selected graphic structure file, in this exemplary embodiment a first graphic structure file 103, is dynamically integrated into the editor program 20 107.

After integration into the editor program 107, a set 111 of selectable elements 112, 113, 114, which are defined in the first graphic structure file 103 as selectable elements for determining a further described graph is displayed to the user 106 on a screen 110 by 25 means of a representation component 109 which is connected to the editor program 107. In addition, in this exemplary embodiment, according to the first graphic structure file 103 a first check program 115 and a second check program 116 are integrated into the 30 editor program 107 and made available to the user 106 for selection.

Each graphic structure file 102, 103, 104, 105 has, in each case, a set of selectable elements for the 35 respective type of graphs, in each case a graphic structure file being

provided for one type of graph. In addition, each graphic structure file 102, 103, 104, 105 may respectively contain a specific check program which is integrated into the respective graphic structure file 102, 103, 104, 105.

After the first graphic structure file 103 has been integrated, elements of the graph are selected by the user 106 and connected to one another so that a graph is determined which is stored in the form of a predefined intermediate language 117 in a memory 118.

In addition, it is symbolically represented in Fig. 1 that the user 106 stores a plurality of structures 119, 120, 121, 122, 123 for describing different graphs, said structures relating to the type of graph predefined by the first graphic file 103.

The first graphic structure file 103 makes available elements which make possible a graph in the form of a Petri net 201 (see Fig. 2).

Fig. 2 shows the representation component 200 which is presented to the user 106 in the form of a screen surface.

The screen surface 200 has a menu list 202 with different selectable options ("File", "Edit", "Object", "View", "Tools", "Settings", "Help").

Menu items are made available to the user by means of individual selectable elements using an immediate access bar 203 by making a single, direct selection of an element.

In addition, a processing bar 204 is represented with selectable options for determining the graph. Thus, in the first graphic structure file 103, a first selection element 205 is provided with which it is possible to select and process objects represented on the screen.

35

The selection and processing of specific elements for a Petri net 201 is made available to the user 106 by means of a set 206 of further selector elements which will be explained in more detail below.

5       A second selector element 207 is described by means of an empty rectangle and symbolizes a time-specific transition.

10       A third selector element 208 symbolizes a timeless transition, which is represented as a selected transition element 220, 221 and 222 in the Petri net 201.

      A fourth selector element 209 symbolizes an edge which is a directed edge in this exemplary embodiment.

15       A fifth selector element 210 symbolizes a forbidden edge which is designated in accordance with the structure rules of a Petri net 201.

20       A sixth selector element 211 symbolizes a place, in each case a place element 223, 224, 225, 226 being represented in the Petri net 201. The place elements 223, 224, 225 and 226 are connected to the transition elements 220, 221, 222 via edges 227, 228, 229, 230, 231 and 232.

25       A seventh selector element 212 symbolizes the possibility of combining a plurality of elements of the Petri net to form a composite element.

      An eighth selector element 213 symbolizes an input of the Petri net 201 and a ninth selector element 214 symbolizes an output of a Petri net 201.

30       The edges and the individual nodes, that is to say the elements of the Petri net 201, are assigned textual information 251, 252, 253, 254, 255, 256, 257, 258, 259, 260 and 261.

In this way it is possible to assign an additional textual description to the individual elements.

5 If a second graphic structure file 104 is integrated into the editor program 107, the second graphic structure file 104 making available elements of an electronic circuit, and thus a graph of an electronic circuit, a screen mask represented in Fig. 3 with a set of selector elements set up for the circuit simulation is produced.

10 The same designations are used in Fig. 3 for the same elements displayed on the screen as represented in Fig. 2.

15 A set 301 of selector elements which are specifically for describing a graph of an electronic circuit contain

- a tenth selector element 302 which symbolizes an electronic resistor,
- an eleventh selector element 303 which symbolizes an electronic capacitor,
- 20 • a twelfth selector element 304 which symbolizes an inductor,
- a thirteenth selector element 305 symbolizing a transistor,
- a fourteenth selector element 306 symbolizing  
25 an operational amplifier,
- a fifteenth selector element 307 symbolizing a non-directed edge, and
- a sixteenth selector element 308 symbolizing a power source.

30 An electronic circuit 110 is determined by the user 106 and has, in this exemplary embodiment, a power source 311, electronic resistors 312, 313, electronic capacitors 314 and 315 and an operational amplifier 316



which are each connected to one another by means of edges 317. In addition, a ground terminal 318 is illustrated in Fig. 3. The individual circuit elements are assigned textual information 319, 320, 321, 322, 5 323, 324, 325, 326 for further explaining the electronic circuit 310.

Fig. 4 shows the method in its method steps in order to clarify the method.

10 In a first step (step 401) a graphic structure file 102, 103, 104, 105 is selected from a set 101 of graphic structure files 102, 103, 104, 105.

15 In a second step (step 402), a selection is made of elements which are available in accordance with the graphic structure file 102, 103, 104, 105 which was selected in the previous step (step 401).

The selected elements are illustrated by the editor program 107 in a further step (step 403).

20 Fig. 5 shows a first computer 500 with a memory 502 and a processor 503 which are each connected to one another by means of a bus 504 and to an input/output interface 501.

The first computer 500 is connected to a screen 505, a keyboard 506 and a computer mouse 507 by means of the input/output interface 501.

25 In addition, the first computer 500 is connected to further computers 510, 520, 530, 540 and 550 via a communications network 560, in the exemplary embodiment an ISDN network (Integrated Services Digital Network).

30 The set 101 of graphic structure files 102, 103, 104, 105 is stored in the first computer 500.

The further computers 510, 520, 530, 540 and 550 each also have a processor 513, 523, 533, 543 and 553 and each have a memory 512, 522, 532, 542 and 552. In each case the processor 513, 523, 533, 543 and 553  
5 and the memory 512, 522, 532, 542 and 552 are connected to the communications network via, in each case, a bus 514, 524, 534, 544 and 554 via an input/output interface 511, 521, 531, 541 and 551. In addition, the further computers 510, 520, 530, 540 and 550 are each  
10 connected to a screen 515, 525, 535, 545 and 555, to a keyboard 516, 526, 536, 546 and 556 and to a computer mouse 517, 527, 537, 547 and 557.

An editor program 508, 518, 528, 538, 548, 558 is stored in each computer 500, 510, 520, 530, 540 and  
15 550. A graphic structure file 102, 103, 104, 105 is selected by a user of a further computer 510, 520, 530, 540 and 550, and requested from the first computer 500 with a request message 570. The first computer 500 transmits the selected graphic structure file 102, 103,  
20 104, 105 in a reply message 580 to the further computer 510, 520, 530, 540 and 550 requesting the graphic structure file 102, 103, 104, 105.

The requesting further computer 510, 520, 530, 540 and 550 has thus received the requested graphic  
25 structure file 102, 103, 104, 105, and it integrates it into its editor program 518, 528, 538, 548, 558, as described in the first exemplary embodiment.

A number of alternatives to the exemplary embodiments described above are illustrated below:

30 The type of elements which are made available by a graphic structure file is generally freely selectable and

depends only on the respective type of graph to be determined.

The technical system can, for example, also be a piece of technical equipment whose characteristics or structure can be described by the graph.

The editor program and the graph illustrated with the editor program can be used as part of a simulation of the technical system.

[1] Publication available on the Internet on September 2, 1998 at the address:

[2] G. Chiola, G. Franceschinis, R. Gaeta and M. Ribaud, GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets, Performance Evaluation, special issue on Performance Modeling Tools, 24 (1&2), pp. 47 - 68, November 1995

[illegible]

A way of implementing the exemplary described above is given below, written in the programming language C, the implementation being divided into three files:

5

### 1. Initialization file:

```

package interfaces;

import java.io.*;
import java.util.*;
import java.awt.*;

import etc.*;
import elements.*;
import mmi.*;
import tools.*;

public class Initialisierung {
    GraphEditor editor;
    // Der hat die Tokens aus der
    Datei
    StreamTokenizer token;
    // Hier kommen alle erlaubten
    Knoten und Kanten aus der
    // .lgc Datei rein.
    // Die Einträge werden mit den
    Namen der Objekte referenziert
    Hashtable gobjekte;
    // Die aktuelle .lgc Datei
    //String configFile;
    // steht jetzt bei den Einstel-
    lungen
    /**
     * Hier stehen alle Attribute
     drin.
     */
    Hashtable attributNamen;
    /**
     * hier kommen die Einträge für
     das Menü Tools
     * hinein.
     */
    Hashtable tools;

    public Initialisie-
    rung(GraphEditor editor) {
        this.editor = editor;
        gobjekte = new Hashtable();
        attributNamen = new Has-
        htable();
        tools = new Hashtable();
    }

    /**
     * Diese Methode würde die er-
     ste Initialisierungsdatei
     * einlesen für die Einstellun-
     gen der Farben, Schriften...
     * Aber ich darf leider nicht.
     */
    /*
     public void readFirst(String
     name) {
         String configFile = new
         String(name);
         int c;
         //Properties properties = new
         Properties();
         //properties = Sy-
         stem.getProperties();
         //filename = new String("..")
         + proper-
         ties.getProperty("file.separator")
         ) + configFile);
         try {
             File file = new
             File(configFile);
             //FileInputStream in = new
             FileInputStream(file);
             FileReader in = new File-
             Reader(file);
             token = new StreamToken-
             izer(in);

             //Einstellen der Optionen
             für token
             to-
             ken.eolIsSignificant(true);
             token.quoteChar('"');
             //token.quoteChar('\\');
             //token.quoteChar('{');
             token.quoteChar('}');

             //Überlese { und , und ;
             to-
             ken.whitespaceChars('{','(');
             to-
             ken.whitespaceChars(','','(',')');
             to-
             ken.whitespaceChars(';',';');

             boolean fertig = false;
             while (!fertig) {
                 switch
                 (c=token.nextToken()){

```

```

c=token.nextToken();

//System.out.print("Wert2 " + to-
ken.nval);

int g =
(int)token.nval;

c=token.nextToken();

//System.out.println("Wert3 " +
token.nval);

int b =
(int)token.nval;

//System.out.flush();

ueberge-
be(auswahl,r,g,b);
}
break;
}
if
(token.sval.equals("FONTS")) {
while (c != ' '){
{
c=token.nextToken();
if (c == Stre-
amTokenizer.TT_WORD) {
String aus-
wahl = token.sval;

//System.out.print("FONT " + to-
ken.sval);

c=token.nextToken();

String font-
name = token.sval;

//System.out.print(" NAME " + to-
ken.sval);

c=token.nextToken();

String style
= token.sval;

//System.out.print(" STYLE " +
token.sval);

c=token.nextToken();

int size
=(int) token.nval;

ueberge-
be(auswahl,fontname,style,size);

//System.out.println(" SIZE " +
token.nval);

}
}
break;
}

```

```

        if
        {token.sval.equals("SHORTCUTS"))
        {
            while (c != ' ')
            {
                c=token.nextToken();
                if (c == '"')
                {
                    String
                    mpunkt = token.sval;

                    //System.out.print("MENUPUNKT " +
                    token.sval);

                    c=token.nextToken();
                    String icon1
                    = token.sval;

                    //System.out.print("ICON1 " + to-
                    ken.sval);

                    c=token.nextToken();
                    String icon2
                    = token.sval;

                    //System.out.println("ICON2 " +
                    token.sval);

                    edi-
                    tor.getShortcutiveiste().addShortB
                    utton();
                }
            }
            break;
        }
        if
        {token.sval.equals("ACCELERATOR")}
        {
            while (c != ' ')
            {
                c=token.nextToken();
                if (c == '"') {
                    String la-
                    bel = token.sval;

                    //System.out.print("MENUPUNKT " +
                    token.sval);

                    c=token.nextToken();
                    if (c ==
                    StreamTokenizer.TT_WORD) {
                        char cut =
                        token.sval.charAt(0);

                        //System.out.println(" TASTEN " +
                        cut);

                        edi-
                        tor.getMenueleiste().addShortcutT
                        oVector(label, cut);
                    }
                }
            }
        }
    }
}

```

```

        }
        break;
    }
    if
    {token.sval.equals("WINDOWSIZE")}
    {
        c=token.nextToken();
        int x
        =(int)token.nval;

        c=token.nextToken();
        int y
        =(int)token.nval;

        //size.setSize(x,y);
        break;
    }
    if
    {token.sval.equals("WINDOWPOSITIO
    N")) {
        c=token.nextToken();
        int x
        =(int)token.nval;

        c=token.nextToken();
        int y
        =(int)token.nval;

        //location.setSize(x,y);
        break;
    }
    if
    {token.sval.equals("AUTHOR")} {
        c=token.nextToken();
        if (c == '"') {
            Sy-
            stem.out.println("AUTHOR " + to-
            ken.sval);
        }
        break;
    }
    if
    {token.sval.equals("TOOLS")} {
        while (c != ' ')
        {
            c=token.nextToken();
            if (c == '"')
            {
                String pfad
                =new String(token.sval);

                //System.out.println("TOOL " +
                token.sval);
            }
        }
    }
}

```

```

c=token.nextToken();
        String fi-
leName =new String(token.sval);
//System.out.println("TOOL " +
token.sval);
c=token.nextToken();
        String text
=new String(token.sval);
//System.out.println("TOOL " +
token.sval);
        edi-
tor.getMenueleiste().addToolToVec
tor(pfad, fileName, text);
    }
    }
    break;
} else
    break;

default:
}
}

in.close();
System.out.flush();
System.out.println("EINLESEN
DER DATEI " +configFile + "
FERTIG");

} catch
(FileNotFoundException e) {
    System.err.println( con-
figFile + " is not found");
} catch (IOException e) {
    e.printStackTrace();
}
} //read first
*/

/**
 * Diese Methode liest eine
 * Toolbar ein.
 * Sie benötigt den Pfad zur
 * Datei und den Dateinamen.
 */
public void readSecond(String
lgcPath, String datei) {
    String configFile = new
String(lgcPath + datei);
    int c;
    try {
        File file = new
File(configFile);
        FileReader in = new File-
Reader(file);
        token = new StreamTokeni-
zer(in);

```

```

//Einstellen der Optionen
für token
to-
ken.eolIsSignificant(false);
token.quoteChar('"');
//token.quoteChar('\\');
//token.quoteChar('{');
token.quoteChar('}');

//Überlese ( und , und ;
to-
ken.whitespaceChars('{','(');
to-
ken.whitespaceChars(','','(',')');
to-
ken.whitespaceChars(';',';');

boolean fertig = false;
while (!fertig) {
    switch
(c=token.nextToken()){
        case StreamTokeni-
zer.TT_EOF:
            fertig= true;
            break;
        case StreamTokeni-
zer.TT_WORD:
            if
(token.sval.equals("TOOLBAR")) {
                Sy-
stem.out.println("Lese Toolbar");
                readTool-
bar(lgcPath);
                break;
            }
            if
(token.sval.equals("MENU")) {
                Sy-
stem.out.println("Lese Menue");
                readMenu();
                break;
            }
            if
(token.sval.equals("ANALYSISBAR"))
            {
                Sy-
stem.out.println("Lese Analyse-
Bar");
                readAnalyse();
                break;
            }
            if
(token.sval.equals("SHORTCUTS"))
            {
                Sy-
stem.out.println("Lese Short-
cuts");
                readShorts();
                break;
            }

```



```

        }
        if
        (token.sval.equals("ACCELERATOR"))
        {
            Sy-
            stem.out.println("Lese Accelera-
            tor");
            readAccel();
            break;
        }
        default:
    }
}

```

```

        in.close();
        System.out.flush();
        System.out.println("EINLESEN
        DER DATEI " + configFile + "
        FERTIG!");
        //und wichtig für die Anzei-
        ge:
        setLayer();
        setAttributNames();
    } catch
    (FileNotFoundException e) {
        System.err.println( configFile + " is not found");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

private void readToolbar(String
lgcPath) {
    int c = '(';
    gobjekte.clear();
    //System.out.println("Jetzt
    kommt die Toolbar");
    try {
        while (c != ')') {
            switch
            (c=token.nextToken()){
                case StreamTokeniz-
                zer.TT_WORD:
                    if
                    (token.sval.equals("NODE")) {
                        //System.out.println("Lese Kno-
                        ten");
                        readNode(lgcPath);
                        break;
                    }
                    if
                    (token.sval.equals("EDGE")) {
                        //System.out.println("Lese Kan-
                        te");
                        readEdge(lgcPath);
                        break;
                    }
                    default:
            }
        }
    }
}

```

```

        //c=token.nextToken();
        //System.out.println("IN
        der TOOLBAR " + c );
    }
    } catch (IOException e) {
        e.printStackTrace();
    }
    //System.out.println("Fertig
    Toolbar");
}

```

```

private void readNode(String
lgcPath) {
    int c = '(';
    String typename = new
    String();
    String image = new String();
    Vector ecken = new Vector();
    Vector konnektoren = new Vec-
    tor();
    Vector konnektorNamen = new
    Vector();
    Attribute attribute = new
    StandardAttribute();
    Color color = new Co-
    lor(255,255,255);
    //System.out.println("Ein
    Knoten");
    try {
        while (c != ')') {
            switch (c){
                case StreamTokeniz-
                zer.TT_WORD:
                    // Wird nicht mehr be-
                    notigt
                    // if
                    (token.sval.equals("TYPE")) {
                        //
                        c=token.nextToken();
                        // Sy-
                        stem.out.println("Lese TYPE" +
                        token.sval);
                        // break;
                        // }
                        if
                        (token.sval.equals("NAME")) {
                            c=token.nextToken();
                            typename = new
                            String(token.sval);
                            // Sy-
                            stem.out.println("Lese NAME" +
                            typename);
                            break;
                        }
                        if
                        (token.sval.equals("ATTRIBUTES")) {
                            attribute = new
                            StandardAttribute();
                        }
                    }
                }
            }
        }
    }
}

```

09785339-030301

```

        while
        ((c=token.nextToken()) != ' ')) {
            String aname =
            new String(token.sval);
            c = to-
            ken.nextToken();
            String wert = new
            String(token.sval);
            attribu-
            te.addAttribute(aname,wert,true);
            attributNa-
            men.put(aname,aname);
            // Sy-
            stem.out.println("Lese Attribut-
            te" + attribute);
        }

        break;
    }
    if
    (token.sval.equals("IMAGE")) {
        c=token.nextToken();
        image = new
        String(token.sval);
        // Sy-
        stem.out.println("Lese IMAGE" +
        image);
        break;
    }
    if
    (token.sval.equals("FILLEDPOLYGON
    ")) {
        ek-
        ken.removeAllElements();
        int x,y;
        while
        ((c=token.nextToken()) != ' ')) {
            x =
            (int)token.nval;

            c=token.nextToken();
            y =
            (int)token.nval;
            ek-
            ken.addElement(new Point(x,y));
            // Sy-
            stem.out.println("Lese POLYGON" +
            ecken);
        }
        // jetzt sollten
        alle Daten da sein, und es
        // kann ein Knoten-
        prototyp erzeugt werden.
        GraphObjekt knoten =
        new FilledPolygonKnoten(typname,
        ecken,
        konnektoren,
        konnektorNamen,

```

```

        attribute);
        kno-
        ten.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
        color);
        // Erzeuge Button
        mit Werkzeug für Werkzeugleiste
        JButton b = new
        JButton(lgcPath + "images/" +
        image,
        typname,
        new KnotenTool(editor,typname),
        editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        // Eintrag in die
        Hashtabelle
        gobjek-
        te.put(typname,knoten);
        // Sy-
        stem.out.println("In Hashtabelle:
        " + gobjekte);
        break;
    }
    if
    (token.sval.equals("POLYGON")) {
        ek-
        ken.removeAllElements();
        int x,y;
        while
        ((c=token.nextToken()) != ' ')) {
            x =
            (int)token.nval;

            c=token.nextToken();
            y =
            (int)token.nval;
            ek-
            ken.addElement(new Point(x,y));
            // Sy-
            stem.out.println("Lese POLYGON" +
            ecken);
        }
        // jetzt sollten
        alle Daten da sein, und es
        // kann ein Knoten-
        prototyp erzeugt werden.
        GraphObjekt knoten =
        new PolygonKnoten(typname,
        ecken,
        konnektoren,
        konnektorNamen,

```

```

attribute);
    kno-
ten.setColor(color);
    // Sy-
stem.out.println("Setze Farbe " +
color);
    // Erzeuge Button
mit Werkzeug für Werkzeugleiste
    // Der Button greift
über den typnamen auf den richti-
gen
    // Knoten zu.
    ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typename,

new KnotenTool(editor,typename),

editor.getToolBar();
    edi-
tor.getToolBar().addToolButton(b)
;
    // Eintrag in die
Hashtabelle
    gobjek-
te.put(typname,knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

    break;
}
if
(token.sval.equals("FILLEDOVAL"))
{
    int breite=10;
    int hoehe=10;
    while
((c=token.nextToken()) != '|') {
        breite =
(int)token.nval;

c=token.nextToken();
        hoehe =
(int)token.nval;
    // Sy-
stem.out.println("Lese OVAL_FILL"
+ token.nval);
    }
    // jetzt sollten
alle Daten da sein, und es
    // kann ein Knoten-
prototyp erzeugt werden.
    GraphObjekt knoten
= new FilledOvalKnoten(typname,

hoehe,

breite,

```

```

konnektoren,

konnektorNamen,

attribute);
    kno-
ten.setColor(color);
    // Sy-
stem.out.println("Setze Farbe " +
color);
    // Erzeuge Button
mit Werkzeug für Werkzeugleiste
    ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typename,

new KnotenTool(editor,typename),

editor.getToolBar();
    edi-
tor.getToolBar().addToolButton(b)
;
    // Eintrag in die
Hashtabelle
    gobjek-
te.put(typname,knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

    break;
}
if
(token.sval.equals("OVAL")) {
    int breite=10;
    int hoehe=10;
    while
((c=token.nextToken()) != '|') {
        breite =
(int)token.nval;

c=token.nextToken();
        hoehe =
(int)token.nval;
    // Sy-
stem.out.println("Lese OVAL" +
token.nval);
    }
    // jetzt sollten
alle Daten da sein, und es
    // kann ein Knoten-
prototyp erzeugt werden.
    GraphObjekt knoten
= new OvalKnoten( typname,

hoehe,

breite,

```

```

    konnektoren,
    konnektorNamen,
    attribute);
        kno-
ten.setColor(color);
        // Sy-
stem.out.println("Setze Farbe " +
color);
        // Erzeuge Button
mit Werkzeug für Werkzeugleiste
        ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typename,
new KnotenTool(editor, typename),

editor.getToolBar());
        edi-
tor.getToolBar().addToolButton(b)
;
        // Eintrag in die
Hashtabelle
        gobjek-
te.put(typename, knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

        break;
    }
    if
(token.sval.equals("CONNECTORS"))
{
        konnekto-
ren.removeAllElements();
        int x,y;
        String name;
        while
((c=token.nextToken()) != '|') {
            x =
(int)token.nval;

c=token.nextToken();
            y =
(int)token.nval;

c=token.nextToken();
            name = to-
ken.sval;
            konnekto-
ren.addElement(new Point(x,y));
            konnektorNa-
men.addElement(name);
            // Sy-
stem.out.println("Lese Konnekto-
ren" + konnektoren);

```

```

        // Sy-
stem.out.println("Die Namen: " +
konnektorNamen);
        }
        break;
    }
    if
(token.sval.equals("COLOR")) {
c=token.nextToken();

//System.out.println("Lese COLOR"
+ token.nval);
        int r =
(int)token.nval;

c=token.nextToken();

//System.out.println("Lese COLOR"
+ token.nval);
        int g =
(int)token.nval;

c=token.nextToken();

//System.out.println("Lese COLOR"
+ token.nval);
        int b =
(int)token.nval;
        color = new Co-
lor(r,g,b);
        break;
    }
    default:
    } //switch
c=token.nextToken();
    // Sy-
stem.out.println("NAECHSTES
TOKEN" + token.sval);
    } //while
    //c=token.nextToken();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // System.out.println("Bende
readNode");

    } //readNode

    private void readEdge(String
lgcPath) {
        // System.out.println("Eine
Kante");
        int c = '|';
        String typename = new
String();
        String image = new String();
        Attribute attribute = new
StandardAttribute();

```

```
c=token.nextToken();  
        winkel =  
(int) token.nval;  
        // Sy-  
stem.out.println("Lese Arrow" +  
radius+ winkel);  
    }  
    // jetzt sollten  
alle Daten da sein, und es  
// kann ein Kanten-  
prototyp erzeugt werden.  
    GraphObjekt kante =  
new PfeilKante(typname,  
  
radius,  
  
winkel,  
  
attribute);  
        kan-  
te.setColor(color);  
        // Sy-  
stem.out.println("Setze Farbe " +  
color);  
        // Erzeuge Button  
mit Werkzeug für Werkzeugleiste  
        ToolButton b = new  
ToolButton(lgcPath + "images/" +  
image,  
  
typname,  
  
new Kantentool(editor, typname),  
  
editor.getToolBar());  
        edi-  
tor.getToolBar().addToolButton(b)  
;        // Eintrag in die  
Hashtabelle  
gobjek-  
te.put(typname,kante);  
  
//System.out.println("In Hashta-  
belle: " + gobjekte);  
  
        break;  
    }  
    if  
(token.sval.equals("POINT")) {  
        int durch = 10;  
        while  
((c=token.nextToken()) != ' ') {  
            durch =  
(int) token.nval;  
            // Sy-  
stem.out.println("Lese Point" +  
durch);  
        }  
        // jetzt sollten  
alle Daten da sein, und es
```

```

        // kann ein Kanten-
        // prototyp erzeugt werden.
        GraphObjekt kante =
        new KreisKante(typname,
        durch,
        attribute);
        kan-
        te.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
        color);
        // Erzeuge Button
        // mit Werkzeug für Werkzeugleiste
        ToolButton b = new
        ToolButton(lgcPath + "images/" +
        image,
        typname,
        new KantenTool(editor, typname),
        editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        // Eintrag in die
        // Hashtabelle
        gobjek-
        te.put(typname, kante);
        //System.out.println("In Hashta-
        //belle: " + gobjekte);
        break;
    }
    if
    (token.sval.equals("NOEND")) {
        while
        ((c=token.nextToken()) != ' ') {
            // durch =
            (int)token.nval;
            // Sy-
            stem.out.println("Lese Point" +
            durch);
        }
        // jetzt sollten
        // alle Daten da sein, und es
        // kann ein Kanten-
        // prototyp erzeugt werden.
        GraphObjekt kante =
        new StandardKante(typname,
        attribute);
        kan-
        te.setColor(color);
        // Sy-
        stem.out.println("Setze Farbe " +
        color);

```

```

        // Erzeuge Button
        // mit Werkzeug für Werkzeugleiste
        ToolButton b = new
        ToolButton(lgcPath + "images/" +
        image,
        typname,
        new KantenTool(editor, typname),
        editor.getToolBar());
        edi-
        tor.getToolBar().addToolButton(b);
        // Eintrag in die
        // Hashtabelle
        gobjek-
        te.put(typname, kante);
        //System.out.println("In Hashta-
        //belle: " + gobjekte);
        break;
    }
    if
    (token.sval.equals("SIZE")) {
        c=token.nextToken();
        // Sy-
        stem.out.println("Lese SIZE" +
        token.nval);
        break;
    }
    if
    (token.sval.equals("COLOR")) {
        //System.out.println("Lese COLOR"
        // + token.nval);
        c=token.nextToken();
        int r =
        (int)token.nval;
        c=token.nextToken();
        //System.out.println("Lese COLOR"
        // + token.nval);
        int g =
        (int)token.nval;
        c=token.nextToken();
        //System.out.println("Lese COLOR"
        // + token.nval);
        int b =
        (int)token.nval;
        color = new Co-
        lor(r,g,b);
        // Sy-
        stem.out.println("Gelesene Farbe:
        " + color);

```

```

        break;
    }
    default:
    } //switch
    c=token.nextToken();
    // Sy-
    stem.out.println("NAECHSTES
    TOKEN" + token.sval);
    } //while
    //c=token.nextToken();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // System.out.println("Bende
    readEdge");
} //readEdge

```

```

private void readMenu() {
    tools.clear();
    int c = '(';
    try {
        while
        ((c=token.nextToken()) != ')') {
            //c=token.nextToken();
            String namen = to-
            ken.sval;
            System.out.println("Jetzt
            kommt das Menu"+ namen);
            c = token.nextToken();
            String aufruf = to-
            ken.sval;
            System.out.println("Jetzt
            kommt das Menu"+ aufruf);
            tools.put(new
            String(namen), new
            String(aufruf));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

private void readAnalyse() {
    System.out.println("Jetzt
    kommt die Analyse");
}

```

```

private void readShorts() {
    System.out.println("Jetzt
    kommt die Shortcut");
}

```

```

private void readAccel() {
    System.out.println("Jetzt
    kommen die Accelerators");
}

```

```

// private void uebergebe
(String mpunkt,String
icon1,String icon2) {
    // public void addBut-
    ton(String menuPunkt, String
    imagel, String image2)
}

```

```

private void uebergebe(String
auswahl,String name,String style,
int size) {
    int styleInt = 0;
    switch (style.charAt(0)){
        case 'B':
            styleInt = Font.BOLD;
            break;
        case 'P':
            styleInt = Font.PLAIN;
            break;
        case 'I':
            styleInt = Font.ITALIC;
            break;
        default:
            styleInt = Font.PLAIN;
    }
}

```

```

Font font = new Font(name,
styleInt, size);
switch (auswahl.charAt(0)){
    case 'M':
        edi-
        tor.getMenuleiste().setFont(font
        );
        break;
    case 'P':
        //noch zu Implementiern
        break;
    case 'S':
        edi-
        tor.getStatusleiste().setFont(fon
        t);
        break;
}
}

```

```

private void uebergebe(String
auswahl,int r,int g,int b) {
    if (auswahl.equals("PAPER")){
        edi-
        tor.getZeichenflaeche().setBackgr
        ound(new Color(r,g,b));
    }
    if (auswahl.equals("GRID")){
        //noch zu implementiern
    }
    if
    (auswahl.equals("MENUBGC")){
        // edi-
        tor.getMenuleiste().setBackgroun
        d(new Color(r,g,b));
    }
    if
    (auswahl.equals("MENUFGC")){
}

```

```

        // menubar.setForeground(new
        Color(r,g,b));
    }
    if
    (auswahl.equals("STATUSBGC")){
        edi-
        tor.getStatusleiste().setBackgrou
        nd (new Color(r,g,b));
    }
    if
    (auswahl.equals("STATUSFGC")){
        edi-
        tor.getStatusleiste().setForegrou
        nd(new Color(r,g,b));
    }
    if
    (auswahl.equals("TOOLBGC")){
        edi-
        tor.getToolbar().setBackground(new
        Color(r,g,b));
    }
    if
    (auswahl.equals("TOOLFGC")){
        edi-
        tor.getToolbar().setForeground
        (new Color(r,g,b));
    }
    if
    (auswahl.equals("SHORTCUTBGC")){
        edi-
        tor.getShortcutleiste().setBackgr
        ound(new Color(r,g,b));
    }
    if
    (auswahl.equals("SHORTCUTFGC")){
        edi-
        tor.getShortcutleiste().setForegr
        ound (new Color(r,g,b));
    }
}

/**
 * Liefert eine Kopie eines
 * GraphObjektes
 * zuruck.
 */
public GraphObjekt getOb-
jekt(String name) {
    if
    (gobjekte.containsKey(name)) {
        GraphObjekt vater =
        (GraphObjekt)gobjekte.get(name);
        return
        (GraphObjekt)vater.copy();
    } else {
        return null;
    }
}

/**
 * Diese Methode fügt alle an-
 * zeigbaren ObjektTypen in die

```

```

 * Hashtable der Klasse Gra-
 * phObjekt ein,
 * -> alle Objekte werden ange-
 * zeigt.
 */
public void setLayer() {
    Hashtable alle = new Has-
    htable(gobjekte.size(),1.0f);
    Enumeration e = gobjek-
    te.keys();
    while (e.hasMoreElements())
    {
        String key =
        (String)e.nextElement();
        alle.put(key,new
        String(key));
    }
    GraphObjekt.toShow = alle;
}

/**
 * Liefert alle anzeigbaren
 * Layers zurück.
 */
public Enumeration getLayers()
{
    return gobjekte.keys();
}

/**
 * Liefert die maximale Anzahl
 * der Layers zurück.
 */
public int countLayers() {
    return gobjekte.size();
}

/**
 * Diese Methode fügt alle an-
 * zeigbaren AttributNamen in die
 * Hashtable der Klasse Attri-
 * bute ein,
 * -> alle Attribute werden an-
 * gezeigt.
 */
public void setAttributNames()
{
    Hashtable alle = new Has-
    htable(attributNamen.size(),1.0f)
    ;
    Enumeration e = attributNa-
    men.keys();
    while (e.hasMoreElements())
    {
        String key =
        (String)e.nextElement();
        alle.put(key,new
        String(key));
    }
    Attribute.toShow = alle;
}

```



```

/**
 * Liefert alle anzeigbaren
 * AttributNamen zuruck.
 */
public Enumeration getAttributNamen() {
    return attributNamen.keys();
}

/**
 * Liefert die maximale Anzahl
 * der Attribute zuruck.
 */
public int countAttributNamen() {
    return attributNamen.size();
}

/**
 * Fugt einen Attribut namen
 * in die

```

```

* Hashtabel ein.
*/
public void addAttributName (
String name) {
    attributNamen.put(new
String(name), new String(name));
}

/**
 *
 */
public Hashtable getTools() {
    return tools;
}

// public String getConfigFile()
{
    // return configFile;
    // }
}

```

## 2."load" file

```

package commands;

import etc.*;
import java.util.*;
import java.awt.*;
import java.io.*;
import interfaces.*;

/**
 * Ladt einen Graphen aus einer
 * .lgf Datei.
 */
public class Load extends Befehl
{
    Vector undo;

    public Load(GraphEditor editor) {
        super(editor);
        undo=new Vector();
        help =
"<filename.lgf/.lgc/.lgt>";
    }

    public void ausfuehren(String[]
param) {
        //System.out.println(param);
        int anzahl = param.length;
        switch (anzahl) {
            case 0 : // bei keinem Ar-
                gument tun wir nichts.
                    break;
            case 1 : // bei einen Ar-
                gument wird erst nachgeschaut!

```

```

        if
(param[0].endsWith(".lgc") ||
        pa-
ram[0].endsWith(".lgf") ||
        pa-
85 ram[0].endsWith(".lgt") ) {
            // wir wurden
            von der CommandoZeile aufgerufen
            File file = new
File(param[0]);

            //System.out.println("Der Pfad :
" + file.getParent());

            //System.out.println("Der Name :
" + file.getName());
            prue-
fe(file.getParent()+"/",file.getN
ame());
        } else {
            //nothing
        }
        break;
        default : //zuviel Parame-
ter
            break;
        } //switch
    }

    public void ausfuehren(String
param) {
        edi-
tor.getStatusleiste().show("Load.
..");

        ((Component)editor).setCursor(Cur

```

```

sor.getPredefinedCursor(Cursor.WAIT_CURSOR));
    FileDialog fd = new FileDialog((Frame)editor, null, FileDialog.LOAD);
    // das hat leider noch keine Auswirkungen in Windows und Solaris
    // ab 1.1.6 gehts doch
    fd.setDirectory(System.getProperty("user.dir"));
    // das schon
    fd.setFile("noname.lgf");
    FilenameFilter filter = new lgFilter();
    fd.setFilenameFilter(filter);
    fd.show();
    String dir = fd.getDirectory();
    String file = fd.getFile();
    // fd.getFile() liefert null bei Abbruch!
    if (file == null) {
        // nichts zu tun
    }

    ((Component)editor).setCursor(Cursor.getDefaultCursor());
    return;
} else {
    // laden

    //System.out.println(fd.getDirectory());

    //System.out.println(fd.getFile());
    Vector ge-
    loescht=editor.getGraph().removeAll();
    pruefe(dir, file);
    edi-
    tor.getGraph().setChanged(false);
    editor.setAuswahl(new Vector());
    Vector lastCommands = editor.getLastCommands();
    if (lastCommands.size() < 10) {
        lastCom-
        mands.addElement(this);
    } else {
        lastCom-
        mands.removeElementAt(0);
        lastCom-
        mands.addElement(this);
    }
    if (undo.size() < 10) {
        undo.addElement(geloescht);
    } else {
        undo.removeElementAt(0);
        undo.addElement(geloescht);
    }
    //else
    edi-
    tor.getZeichenflaeche().drawBuffer(editor.getGraph());

    ((Component)editor).setCursor(Cursor.getDefaultCursor());
    edi-
    tor.getStatusleiste().show("Done");
}

/**
 * Macht Datei laden rückgan-
 * g.
 */
public void undo() {
    edi-
    tor.getStatusleiste().show("Undo: Load...");

    ((Component)editor).setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
    if (!undo.isEmpty()) {
        Vector insert = (Vector)undo.lastElement();
        if (insert != null) {
            edi-
            tor.getGraph().removeAll();
            edi-
            tor.getGraph().add(insert);
            insert.removeAllElements();
        }

        undo.removeElement(undo.lastElement());
    }
    edi-
    tor.getZeichenflaeche().drawBuffer(editor.getGraph());
    edi-
    tor.getGraph().setChanged(true);
    edi-
    tor.getStatusleiste().show("Done");

    ((Component)editor).setCursor(Cursor.getDefaultCursor());
    //undo

    /**
     * Wiederholt Datei laden..
     */
    public void redo() {
        edi-
        tor.getStatusleiste().show("Redo: Load...");
        ausfuehren();
    }
}

```

```

    } // redo

    /**
     * Diese Klasse wird leider
     * nicht an
     * die Windows bzw Solaris Kom-
     * ponente
     * weitergereicht.
     */
    class lgFilter implements Fi-
    lenameFilter {
        public boolean accept (File
        dir, String name) {
            return ( na-
            me.endsWith(".lgf") ||
                na-
            me.endsWith(".lgc") ||
                na-
            me.endsWith(".lgt") );
        }
    }

    /**
     * Diese Methode überprüft, ob
     * die richtige
     * * Konfigurationsdatei geladen
     * ist, ansonsten wird
     * * versucht die richtige zu la-
     * den. (->Editor zurücksetzen)
     * * Dannach wird die gewünschte
     * .lgt oder .lgf Datei
     * * geladen.
     */
    private void pruefe (String
    pfad, String datei) {
        Einstellungen settings= edi-
        tor.getEinstellungen();
        if (datei.endsWith(".lgc")) {
            //System.out.println("eine
            lgc Datei");
            File f = new File(pfad +
            datei);
            if (f.exists()) {
                settings.appName = Ein-
                stellungen.format(datei);
                settings.fileName=" ";
                settings.frameName = set-
                tings.fileName+ " "
                +settings.appName + " "
                +settings.copyright;
                settings.configFile = new
                String(datei);
                settings.lgcPath = new
                String(pfad);
                //wir Starten den Editor
                neu
                editor.start();
            } else {
                System.err.println("File
                not found : "+ settings.lgcPath +
                datei);
            }
        }
    }

```

```

    } else if
    (datei.endsWith(".lgf")) {
        //System.out.println("eine
        lgf Datei");
        File f = new File(pfad +
        datei);
        if (f.exists()) {
            settings.fileName = da-
            tei;
            // wir holen uns noch den
            namen des .lgc Files:
            String config = edi-
            tor.getDateischnittstelle().getCo-
            nfig(pfad + datei);
            //System.out.println("Der
            neue Name der Lgc datei " + con-
            fig);
            f = new
            File(settings.lgcPath + config);
            if (f.exists()) {
                // ist diese lgc Datei
                schon geladen?
                if
                (settings.configFile.equals(config)) {
                    //wir muessen nur die
                    lgf Datei laden
                    edi-
                    tor.getDateischnittstelle().load(
                    pfad,datei,editor.getGraph());
                    settings.frameName =
                    settings.fileName+ " "
                    +settings.appName + " "
                    +settings.copyright;
                    ((Frame)editor). set-
                    Title(settings.frameName);
                } else {
                    // wir müssen auch
                    die Konfigurationsdatei laden
                    settings.appName =
                    Einstellungen.format(config);
                    settings.configFile =
                    new String(config);
                    settings.frameName =
                    settings.fileName+ " "
                    +settings.appName + " "
                    +settings.copyright;
                    //wir Starten den
                    Editor neu
                    editor.start();
                    edi-
                    tor.getDateischnittstelle().load(
                    pfad,datei,editor.getGraph());
                }
            } else {
                Sy-
                stem.err.println("File not found
                : " + settings.lgcPath + config);
            }
        } else {
            System.err.println("File
            not found : " + pfad + datei);
        }
    }

```

```

    }
    //start();
    } else if
    (datei.endsWith(".lgt")) {
        //System.out.println("eine
        lgt Datei");
        File f = new File(pfad +
        datei);
        if (f.exists()) {
            settings.fileName = da-
            tei;
            settings.frameName = set-
            tings.fileName+ " "
            +settings.appName + " "
            +settings.copyright;
            // wir holen uns noch den
            namen des .lgc Files:
            //String config = edi-
            tor.getDateischnittstelle().getCo
            nfig(pfad + datei);
            //System.out.println("Der
            neue Name der Lgc datei " + con-
            fig);
            //f = new
            File(settings.lgcPath + config);
            //if (f.exists()) {
            // ist diese lgc Datei
            schon geladen?
            //if
            (settings.configFile.equals(confi
            g)) {
                //wir muessen nur die
                lgt Datei laden und interpretie-
                ren
                LgtInterpreter inter-
                preter=editor.getInterpreter();
            //System.out.println("Der Inter-
            preter : " + interpreter);
            if (interpreter ==
            null) {
                interpreter = new
                LgtInterpreter(editor, pfad + da-
                tei);
                edi-
                tor.setInterpreter(interpreter);
                interpre-
                ter.start();
            } else {

```

```

                interpreter
                ter.setFile(pfad + datei);
            }

            //Dateischnittstelle().load(pfad,
            datei, editor.getGraph());
            //settings.frameName
            = settings.appName + " " + set-
            tings.fileName;
            //((Frame)editor).
            setTitle(settings.frameName);
            // } else {
            // wir muessen auch
            die Konfigurationsdatei laden
            // settings.appName =
            Einstellungen.format(config);
            //settings.configFile
            = new String(config);
            //settings.frameName
            = settings.appName + " " + set-
            tings.fileName;
            //wir Starten den
            Editor neu
            //editor.start();
            // LgtInterpreter in-
            terpreter = new LgtInterpre-
            ter(editor, pfad + datei);
            // edi-
            tor.setInterpreter(interpreter);
            // interpre-
            ter.start();
            // }
            //} else {
            // Sy-
            stem.err.println("File not found
            : " + settings.lgcPath + config);
            //}
            } else {
                System.err.println("File
                not found : " + pfad + datei);
            }
            } else {
                System.err.println("usage:
                java LoGraph2 <path to config-
                files> AND <file.lgc> OR
                <file.lgf> OR <file.lgt>");
            }
        }
    }
}

```

### 3. "toolbar" file

```

package mmi;

import java.awt.*;
import java.awt.event.*;

import etc.*;
import tools.*;

```

```

/**
 * Über das aktuelle Tool der
 * Toolbar werden die
 * Maus Aktionen des Benutzers an
 * den Graphen weitergegeben.

```

```

    * Die Toolbar ermöglicht das
    hinzufügen und entfernen
    * von ToolButtons, und deren zu-
    gehörigen ActionListener.
    */

```

```

public class Toolbar extends Pa-
nel {
    GraphEditor editor;
    Tool currentTool;
    ToolButton currentButton;
    int borderSize = 4;
    /**

```

```

        * Der Konstruktor erzeugt das
        AuswahlTool,
        * da dieses immer vorhanden
        sein sollte.
        */

```

```

        public Toolbar(GraphEditor edi-
        tor){
            this.editor = editor;
            setLayout(new BarLay-
            out(BarLayout.VERTICAL,2));
            setBackgroundCo-
            und(editor.getEinstellungen().too-
            lbarBgCo);
            // eine kleine Lucke
            add(new Space(5,24));

```

```

            ToolButton b = new ToolBut-
            ton(editor.getEinstellungen().lge
            Path +

```

```

            "images/auswahl.gif",

```

```

            "Select",

```

```

            new AuswahlTool(editor),this);
            setCurrentTool(b.getTool());
            setCurrentButton(b);
            add(b);
            add(new Space(5,24));
        }

```

```

        public Insets getInsets() {
            Insets insets =
            (Insets)(super.getInsets()).clone
            ();
            insets.top += borderSize;
            insets.left +=
            (borderSize+2);
            insets.bottom += borderSize;
            insets.right +=
            (borderSize+2);
            return insets;
        }

```

```

        public void paint(Graphics g) {
            super.paint(g);
            Insets insets = su-
            per.getInsets();

```

```

            int w = getSize().width-
            insets.left-insets.right;
            int h = getSize().height-
            insets.top-insets.bottom;

```

```

            g.setColor(editor.getEinstellunge-
            n().toolbarBgCo);
            for (int i=0; i<borderSize;
            i++) {

```

```

                g.draw3DRect(i+insets.left,i+inse-
                ts.top,
                                w-2*i-1, h-
                2*i-1, i<borderSize/2);
            }

```

```

            /**
            * Fügt einen ToolButton hinzu.
            */

```

```

            public void addToolBut-
            ton(ToolButton button) {
                add(button);
            }

```

```

            /**
            * Entfernt einen ToolButton.
            */

```

```

            public void deleteTooleBut-
            ton(ToolButton button) {

```

```

            /**
            * Setzt das aktuelle Tool;
            * wird normalerweise von den
            ToolButtons aufgerufen.
            */

```

```

            public void setCurrentTool(Tool
            currentTool) {
                this.currentTool = current-
                Tool;
                this.currentTool.reset();
            }

```

```

            /**
            * Setzt den aktuellen Button,
            damit der nächste
            * aktuelle Button ihn zurück-
            setzen kann.
            */

```

```

            public void setCurrentBut-
            ton(ToolButton currentButton) {
                if (this.currentButton !=
                null)
                    this.currentButton.setUp();
                this.currentButton = current-
                Button;
                this.currentButton.setDown();
            }

```

```

            /**

```

```
    * Liefert das aktuelle Tool
    zurück.
    * wird normalerweise von den
    Zeichenfläche aufgerufen.
    */
    public Tool getCurrentTool() {
        return currentTool;
    }

    /**
    * Liefert den aktuellen But-
    ton, damit der nächste
    * aktuelle Button ihn zurück-
    setzen kann.
    */
```

```
    public ToolButton getCurrent-
    Button() {
        return currentButton;
    }

    /**
    * Liefert den Editor an die
    Buttons weiter.
    */
    public GraphEditor getEditor()
    {
        return editor;
    }
} //Toolbar
```

**Patent claims**

1. A method for determining a graphic structure of a technical system,
- 5 a) in which a graph structure file is selected from a set of a plurality of different graph structure files, a graph structure file containing in each case indications of which elements can be selected to represent it in order to describe the structure of the
- 10 technical system graphically,
- b) in which elements are selected in such a way that the technical system is described using the selected elements, and
- c) in which the elements are represented by an editor
- 15 program into which the selected graph structure file has been integrated, by which means the graphic structure of the technical system is determined.
2. The method as claimed in claim 1, in which the technical system is an electronic circuit.
- 20 3. The method as claimed in claim 2, in which the technical system is a piece of technical equipment.
4. The method as claimed in one of claims 1 to 3, in which the elements are graph elements of a graph which describe the technical system.
- 25 5. The method as claimed in one of claims 1 to 4, in which the graphic structure of the technical system which is determined is checked for predefined structure rules.
6. An arrangement for determining a graph
- 30 structure of a technical system,
- a) having a memory in which a set of a plurality of different graph structure files are stored,

-34 -

a graph structure file containing in each case indications of which elements can be selected to represent it in order to form a graph,

5 b) having a selector unit with which a graph structure file can be selected from the set of graph structure files,

c) having a processor which is configured in such a way that an editor program can be executed, with which editor program a graph structure file selected from the  
10 set of graph structure files can be used to determine a graph with elements of the selected graph structure file, by which means the graph structure is determined,

d) having a representation component which is coupled to the editor program and with which the specific graph  
15 structure can be represented.

7. The arrangement as claimed in claim 6, in which a structure of a technical system is described using the graph.

8. The arrangement as claimed in claim 7, in which  
20 the technical system is an electronic circuit.

9. The arrangement as claimed in claim 7, in which the technical system is a piece of technical equipment.

10. The arrangement as claimed in claim 6,

a) having a first subarrangement which has the  
25 memory,

b) having a second subarrangement which is coupled to the first subarrangement and has the following components:

- the selector unit,
- 30 - the editor program,
- the representation component.



11. The arrangement as claimed in claim 10, in which the first subarrangement and the second subarrangement are coupled to one another by means of a communications network.
- 5 12. The set of arrangements as claimed in claim 10 or 11, in which a structure of a technical system is described using the graph.
13. The arrangement as claimed in claim 12, in which the technical system is an electronic circuit.
- 10 14. The arrangement as claimed in claim 12, in which the technical system is a piece of technical equipment.

**Abstract**

**Method for determining a graphic structure of a technical system and arrangement and set of arrangements for determining a graph structure**

A graph structure file is selected from a set of a plurality of different graph structure files. A graph structure file contains in each case indications of which elements can be selected to represent it in order to describe the structure of the technical system graphically. Elements are selected in such a way that the selected elements describe the technical system, and the elements are represented by an editor program into which the selected graph structure file has been integrated.

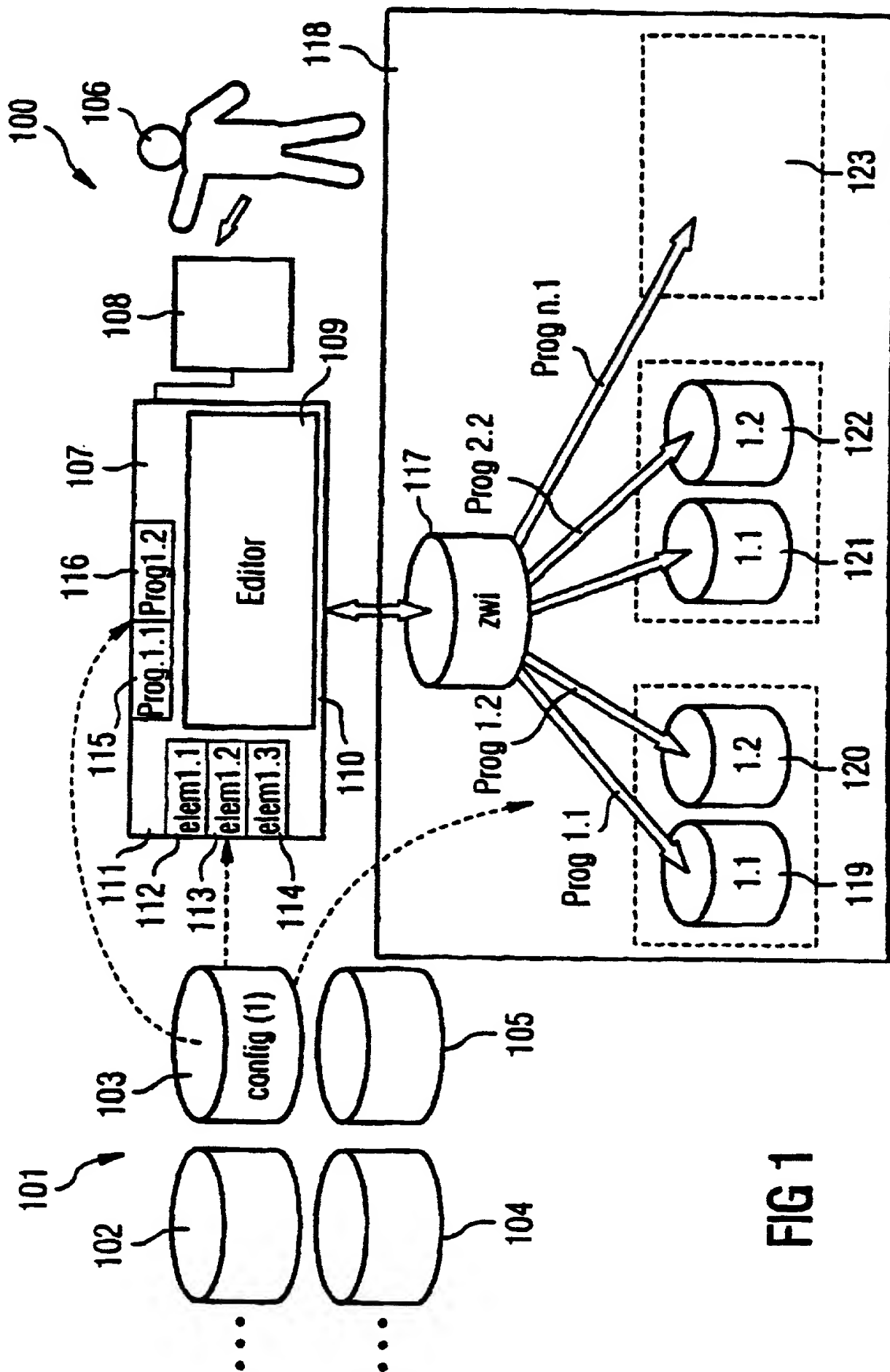
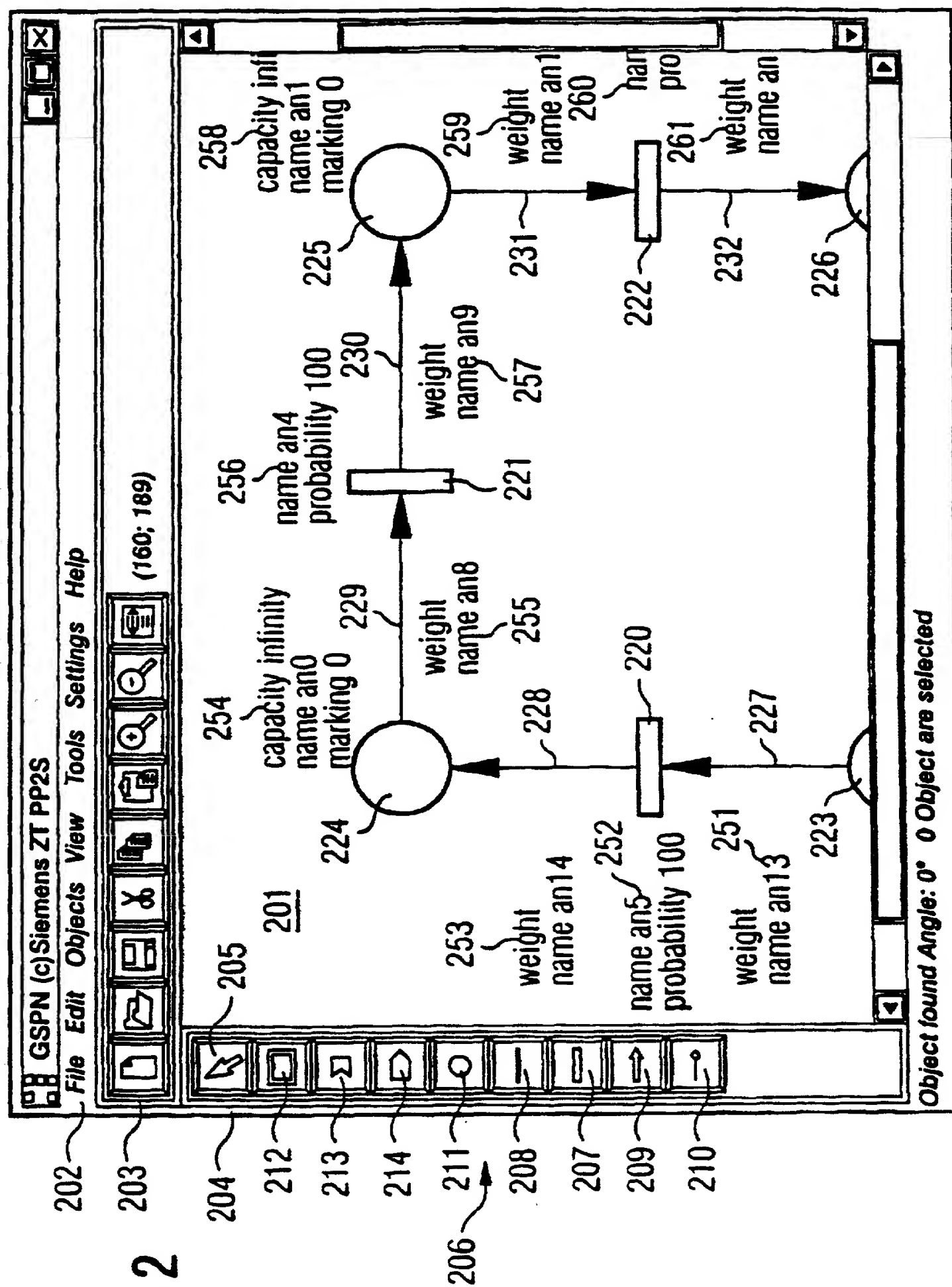
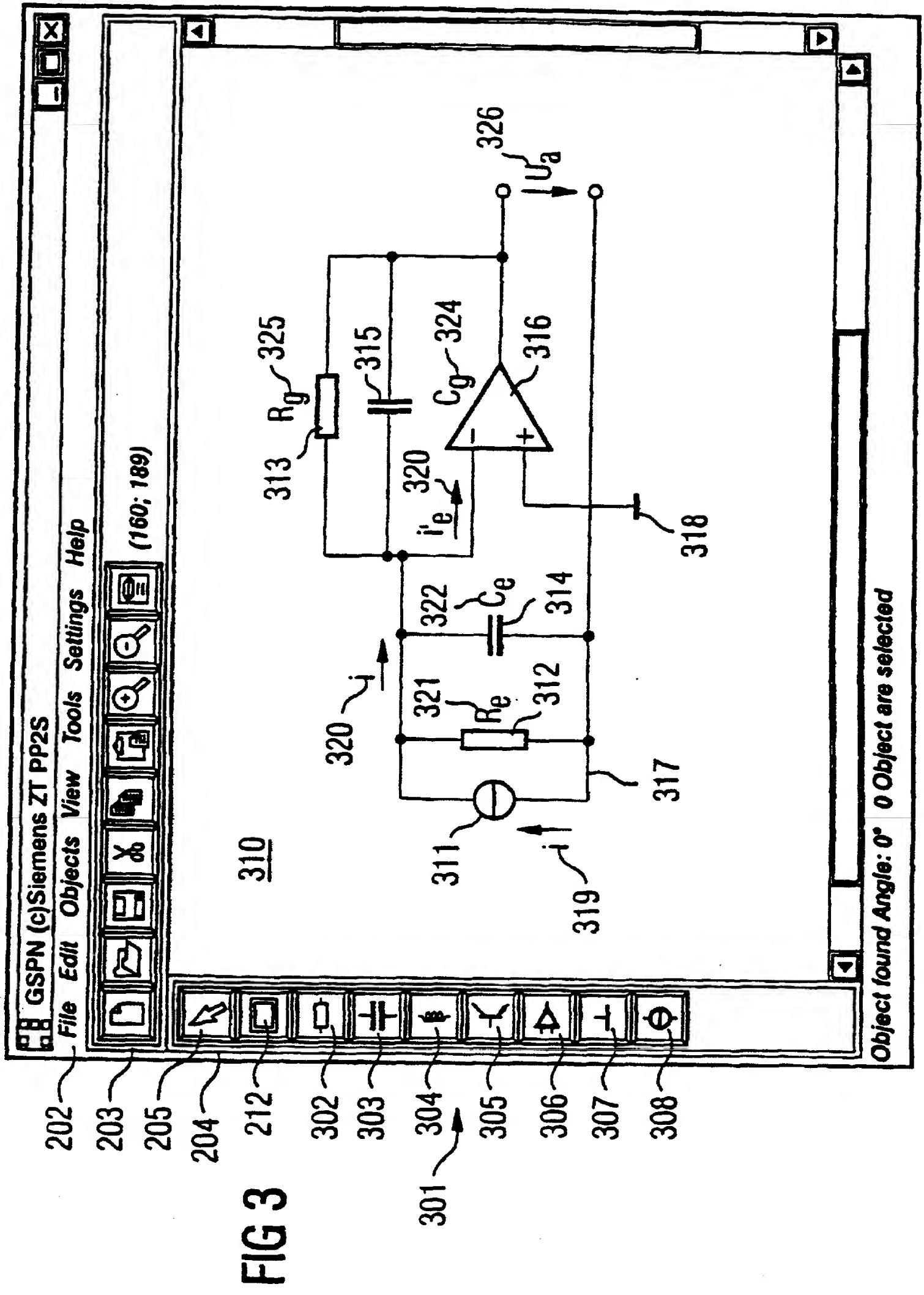


FIG 1





4/5

## FIG 4

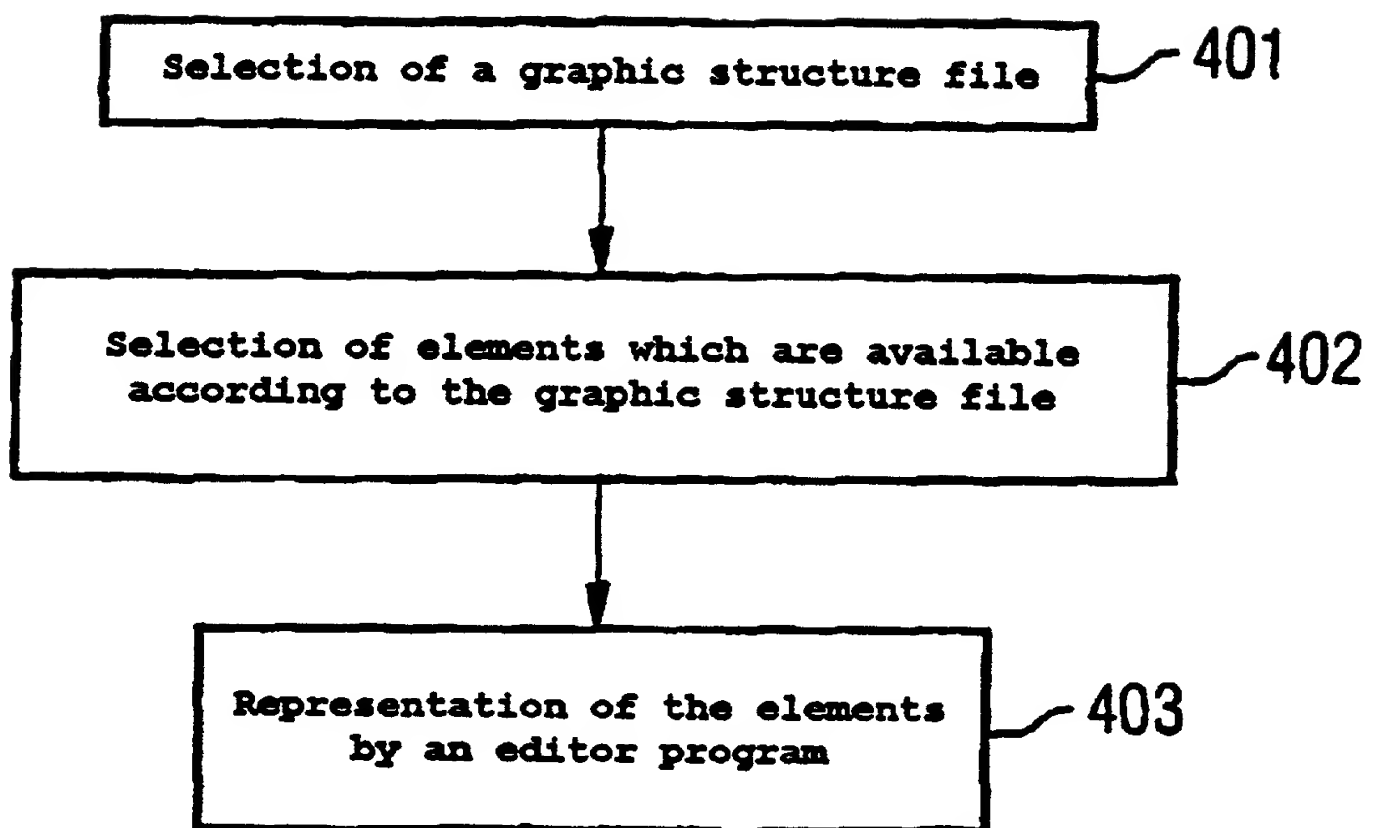
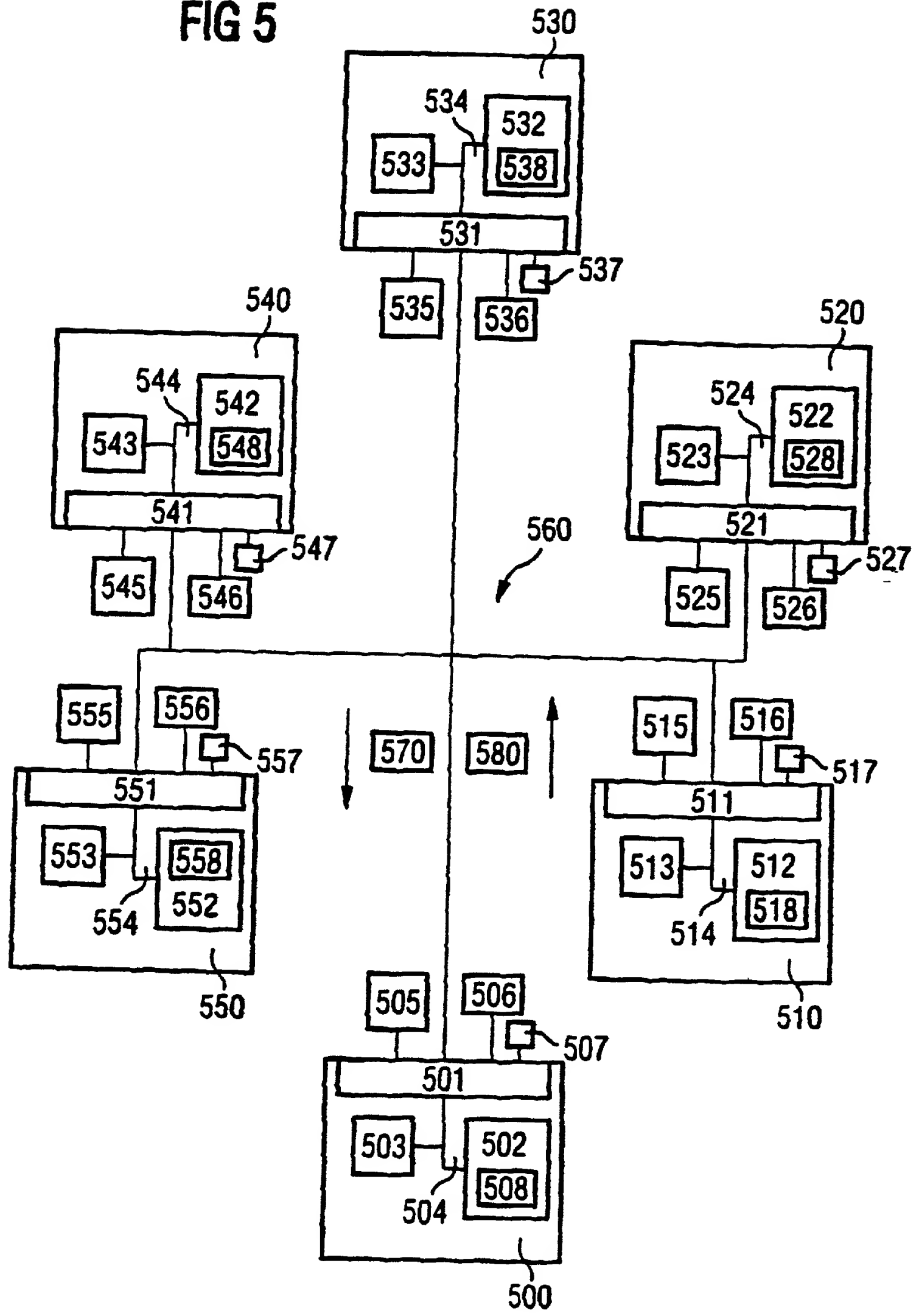


FIG 5



**DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION**  
**ERKLÄRUNG FÜR PATENTANMELDUNGEN MIT VOLLMACHT**  
**German Language Declaration**

Als nachstehend benannter Erfinder erkläre ich hiermit  
an Eides Statt:

dass mein Wohnsitz, meine Postanschrift, und meine  
Staatsangehörigkeit den im Nachstehenden nach  
meinem Namen aufgeführten Angaben entsprechen,

dass ich, nach bestem Wissen der ursprüngliche, erste  
und alleinige Erfinder (falls nachstehend nur ein Name  
angegeben ist) oder ein ursprünglicher, erster und  
Miterfinder (falls nachstehend mehrere Namen  
aufgeführt sind) des Gegenstandes bin, für des dieser  
Antrag gestellt wird und für den ein Patent beantragt  
wird für die Erfindung mit dem Titel;

**Verfahren zur Bestimmung Einer Graphischen  
Struktur Eines Technischen Systems und  
Anordnung Sowie Satz Von Anordnungen zur  
Bestimmung Einer Graphen-Struktur**

deren Beschreibung

(zutreffendes ankreuzen)

☒ hier beigelegt ist.

☐ am \_\_\_\_\_ als  
PCT internationale Anmeldung  
PCT Anmeldungsnummer \_\_\_\_\_  
eingereicht wurde und am \_\_\_\_\_  
abgeändert wurde.

Ich bestätige hiermit, dass ich den Inhalt der obigen  
Patentanmeldung einschliesslich der Ansprüche  
durchgesehen und verstanden habe, die eventuell  
durch einen Zusatzantrag wie oben erwähnt  
abgeändert wurde.

Ich erkenne meine Pflicht zur Offenbarung  
irgendwelcher Informationen, die für die Prüfung der  
vorliegenden Anmeldung in Einklang mit Absatz 37,  
Bundesgesetzbuch, Paragraph 1.56 von Wichtigkeit  
sind, an.

Ich beanspruche hiermit ausländische Prioritätsvorteile  
gemäss Abschnitt 35 der Zivilprozessordnung der  
Vereinigten Staaten, Paragraph 119 aller unten  
angegebenen Auslandsanmeldungen für ein Patent  
oder eine Erfindersurkunde, und habe auch alle  
Auslandsanmeldungen für ein Patent oder eine  
Erfindersurkunde nachstehend gekennzeichnet, die  
ein Anmeldedatum haben, das vor dem  
Anmeldedatum der Anmeldung liegt, für die Priorität  
beansprucht wird.

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are  
as stated below next to my name,

I believe I am the original, first and sole inventor (if only  
one name is listed below) or an original, first and joint  
inventor (if plural names are listed below) of the  
subject matter which is claimed and for which a patent  
is sought on the invention entitled

the specification of which

(check one)

☐ is attached hereto

☐ was filed on \_\_\_\_\_ as  
PCT international application  
PCT Application No. \_\_\_\_\_  
and was amended on \_\_\_\_\_

I hereby state that I have reviewed and understand the  
contents of the above identified specification, including  
the claims as amended by any amendment referred to  
above.

I acknowledge the duty to disclose information which  
is material to the examination of this application in  
accordance with Title 37, Code of Federal Regulations,  
§1.56.

I hereby claim foreign priority benefits under Title 35,  
United States Code, §119 of any foreign application(s)  
for patent or inventor's certificate listed below and have  
also identified below any foreign application for patent  
or inventor's certificate having a filing date before that  
of the application on which priority is claimed:



### German Language Declaration

Prior foreign applications  
Priorität beansprucht

188 39 972.3 Germany 02 September 1998  
(Number) (Country) (Day Month Year Filed)  
(Nummer) (Land) (Tag Monat Jahr eingereicht)

\_\_\_\_\_  
(Number) (Country) (Day Month Year Filed)  
(Nummer) (Land) (Tag Monat Jahr eingereicht)

\_\_\_\_\_  
(Number) (Country) (Day Month Year Filed)  
(Nummer) (Land) (Tag Monat Jahr eingereicht)

Priority Claimed

☒ ☐  
Yes No  
Ja Nein

☐ ☐  
Yes No  
Ja Nein

☐ ☐  
Yes No  
Ja Nein

Ich beanspruche hiermit gemäss Absatz 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 120, den Vorzug aller unten aufgeführten Anmeldungen und falls der Gegenstand aus jedem Anspruch dieser Anmeldung nicht in einer früheren amerikanischen Patentanmeldung laut dem ersten Paragraphen des Absatzes 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 122 offenbart ist, erkenne ich gemäss Absatz 37, Bundesgesetzbuch, Paragraph 1.56 meine Pflicht zur Offenbarung von Informationen an, die zwischen dem Anmeldedatum der früheren Anmeldung und dem nationalen oder PCT internationalen Anmeldedatum dieser Anmeldung bekannt geworden sind.

\_\_\_\_\_  
(Application Serial No.)  
(Anmeldeseriennummer)

\_\_\_\_\_  
(Filing Date)  
(Anmeldedatum)

\_\_\_\_\_  
(Application Serial No.)  
(Anmeldeseriennummer)

\_\_\_\_\_  
(Filing Date)  
(Anmeldedatum)

Ich erkläre hiermit, dass alle von mir in der vorliegenden Erklärung gemachten Angaben nach meinem besten Wissen und Gewissen der vollen Wahrheit entsprechen, und dass ich diese eidesstattliche Erklärung in Kenntnis dessen abgebe, dass wissentlich und vorsätzlich falsche Angaben gemäss Paragraph 1001, Absatz 18 der Zivilprozessordnung der Vereinigten Staaten von Amerika mit Geldstrafe belegt und/oder Gefängnis bestraft werden koennen, und dass derartig wissentlich und vorsätzlich falsche Angaben die Gültigkeit der vorliegenden Patentanmeldung oder eines darauf erteilten Patentes gefährden können.

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §122 I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

\_\_\_\_\_  
(Status)  
(patentiert, anhängig,  
aufgegeben)

\_\_\_\_\_  
(Status)  
(patented, pending,  
abandoned)

\_\_\_\_\_  
(Status)  
(patentiert, anhängig,  
aufgegeben)

\_\_\_\_\_  
(Status)  
(patented, pending,  
abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

**German Language Declaration**

**VERTRETUNGSVOLLMACHT:** Als benannter Erfinder beauftrage ich hiermit den nachstehend benannten Patentanwalt (oder die nachstehend benannten Patentanwälte) und/oder Patent-Agenten mit der Verfolgung der vorliegenden Patentanmeldung sowie mit der Abwicklung aller damit verbundenen Geschäfte vor dem Patent- und Warenzeichenamt:

**POWER OF ATTORNEY:** As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

And I hereby appoint all Attorneys identified by United States Patent & Trademark Office customer number 26574, who are all members of the firm of Schiff Hardin and Waite.

Telefongespräche bitte richten an:  
(Name und Telefonnummer)

Direct Telephone Calls to: (name and telephone number)

312/258-5500

Postanschrift:

Send Correspondence to:  
**SCHIFF HARDIN & WAITE**  
**PATENT DEPARTMENT**  
6600 Sears Tower, Chicago, Illinois 60606-6473

**CUSTOMER NUMBER 26574**

Voller Name des einzigen oder ursprünglichen Erfinders: <b>THURNER, Erwin</b>		Full name of sole or first inventor:	
Unterschrift des Erfinders <i>Erwin Thurner</i>	Datum <i>1.3.2001</i>	Inventor's signature	Date
Wohnsitz <b>82152 Planegg, Germany DEX</b>		Residence	
Staatsangehörigkeit <b>Bundesrepublik Deutschland</b>		Citizenship	
Postanschrift <b>Josef-V-Hirsch-Str. 23</b>		Post Office Address	
<b>82152 Planegg</b>			
<b>Bundesrepublik Deutschland</b>			

Voller Name des zweiten Mitfinders (falls zutreffend):		Full name of second joint inventor, if any:	
Unterschrift des Erfinders	Datum	Inventor's signature	Date
Wohnsitz		Residence	
Staatsangehörigkeit		Citizenship	
Postanschrift		Post Office Address	

(Bitte entsprechende Informationen und Unterschriften im Falle von weiteren Mitfindern angeben).

(Supply similar information and signature for subsequent joint inventors).